欧姆社学习漫画

漫画数据库

〔日〕高桥麻奈 / 著
〔日〕あづま笙子 / 漫画绘制
〔日〕株式会社TREND-PRO / 漫画制作
崔建锋 / 译

科学出版社
www.sciencep.com

✿前　言✿

目前，在计算机的各种应用系统中，数据库已经成为不可或缺的一部分。购买本书的人当中可能有考虑日常业务中引进数据库的，也可能有需要开发用到数据库的业务系统的。虽然数据库是支持这些系统的幕后技术，但我们对它的真正内涵还不太理解。

通过本书，读者可以一边阅读漫画故事一边学习数据库的基础知识。在各章的故事后面，本书还为大家准备了拓展性的知识和用于复习的练习题。练习题对于信息处理技术员考试也是很有帮助的。在确认对各章知识的理解的同时，大家也能够逐渐掌握数据库技术。

本书由以下章节构成：

第 1 章，学习数据库的必要性。为什么需要数据库？没有数据库的话会带来什么不方便？学习数据库的必要性背景。

第 2 章，学习数据库的基本术语。掌握数据库相关的不为人熟知的术语。

第 3 章，学习数据库的设计方法。主要学习作为当今主流的关系数据库的设计方法。

第 4 章，学习处理关系数据库的 SQL 命令。学会 SQL 能够让我们轻松地处理数据库中的数据。

第 5 章，学习数据库的构造。数据库是一个许多人共享的系统。在本章我们将学习为什么通过数据库系统能够共享数据，其架构原理是什么。

第 6 章，学习数据库的应用。主要学习在 Web 等系统中实际使用的数据库系统。

本书的出版得益于众多人士的共同努力。感谢绘制漫画的笙子小姐，制作漫画的 TREND-PRO 公司，负责本书策划、编辑、发行的欧姆社的各位同仁。感谢所有相关工作人员。

衷心地希望本书能对读者有所帮助。

高桥麻奈

✿目　录✿

第 4 章　使用数据库——SQL 的基本操作　85

第 5 章　数据库的应用　129

第 6 章　数据库的普及和灵活应用　177

第1章

什么是数据库

为什么数据库非常必要

嗯——

露娜公主！

どーん
工作一点都没有减少！

被誉为水果之国的编码王国……
大地的恩惠
哈哈哈
今年大丰收！

忙碌应该是一件高兴的事啊！！
啪

唉
要是能更有效率就好了！

关于国内生产的出口水果，
商品部
外国部
出口部
王室分别设置了"商品部"、"外国部"和"出口部"，每个部门都需要分别做文件进行管理。
嗯！

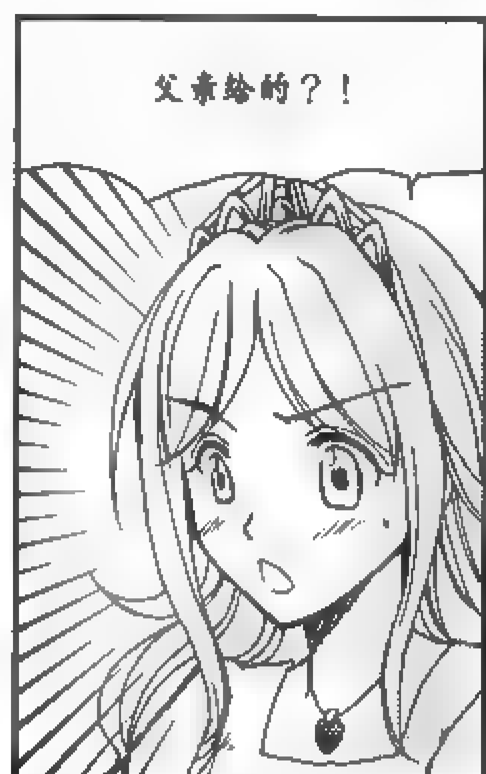
这是非常出色高效的系统呦！
好吧！！工作，工作！！
啦啦啦~~~~
噜噜~~
吧嗒

嗯!

水果和销售由每个部门管理，总觉得有点浪费。前段时间苹果价格上涨，真的有很多工作要做……

露娜公主！

是凯恩吗……
这次又是什么？
国王带回来的东西….


父亲给的？！

那两个人怎么就不在城里好好呆着呢……
公主?
真的要去吗？
ふるふる

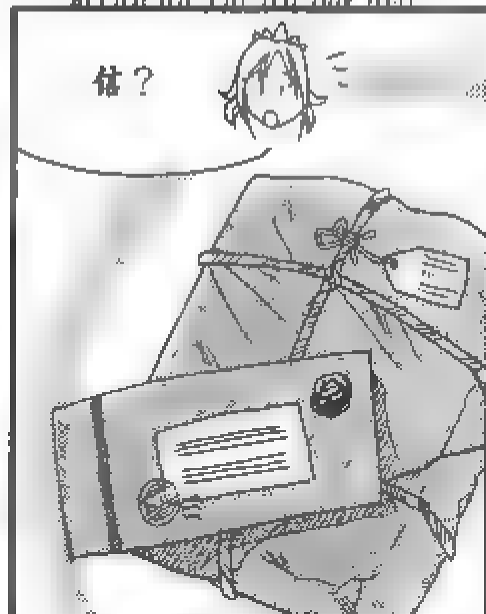
嗯，出访也是非常重要的公务之一！
哈哈哈！

凯恩，出访期间露娜就拜托给你了！
是，是！
再见！

保重哟！

哈 哈 哈
真是的！！
把工作交给我，自己却去旅行！！
再见！
别生气……

这也是对露娜的信任嘛。
对了，那是什么啊？

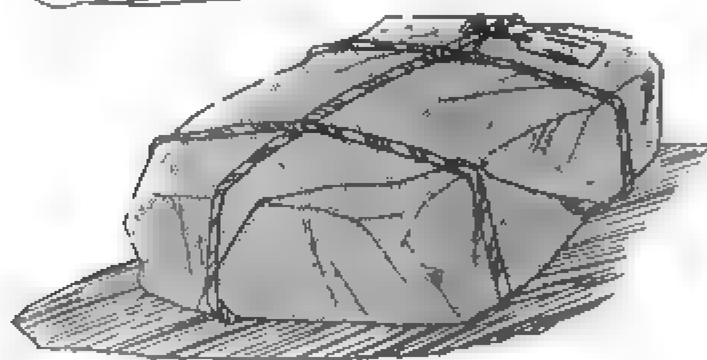
什么呢……包裹和……
一封信。

信？

在出访的一个国家里，我们得到了一本具有划时代意义的技术书……

据给我们书的人说，这本书讲了一个叫做“数据库”(database)的神秘技术……
数据库好像是一个能够让大家共享数据，并可以共同管理和使用数据的便利系统……
但是，如何使用它还要看打开的人，
那个人相信，如果是和谐的编码王国，就一定会和平使用它，所以把书赠给了我们。
露娜！

让我利用你吧！！
却不懂人家的心情……

打开这本书，把它用于我们国家的事业开展吧！



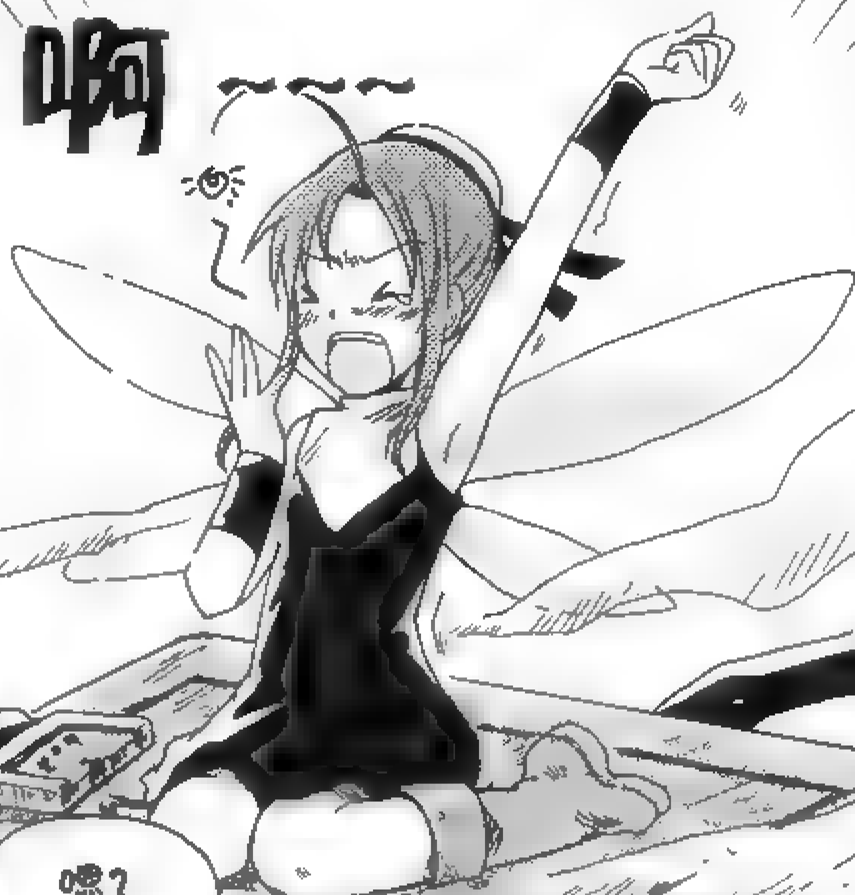
什么呀！！

哇，是一本古书……

咔

打不开啊
上锁了！
用这个吧，
信封中找到的……

ぼぉおおおおん
哇呀！！
！！！
妖，妖怪。



啊~~~
嗯？
这是哪儿啊？
你们是谁？
这里是编码城！
糊，糊涂了……
我叫凯恩！！
是公主的随从……
你是谁……
飞起来了！？
妖……
妖怪……
呀啊啊啊

气呼呼

真没礼貌！！

我是小T，是个精灵！

精灵……

才不是妖怪呐！！

但是只有打开书的人能看到，好像和妖怪也差不多……

嘿嘿

哇啊啊

好像没有害处……
暂且……

嗯……

当然不会有危险了！
露娜和凯恩是想要了
解数据库的知识，

才打开这本书的吗？

嗯，是这样……

那就赶快！！

为了构筑
数据库……

等，
等一下！！

首先，

数据库是什么呀？

啊？居然不知道啊？

现在你们不是在处理各
种各样的数据和数字吗？

是……
但是有很多
问题……

"商品"、"顾客"、"销售"
等数值、数字这些数据，

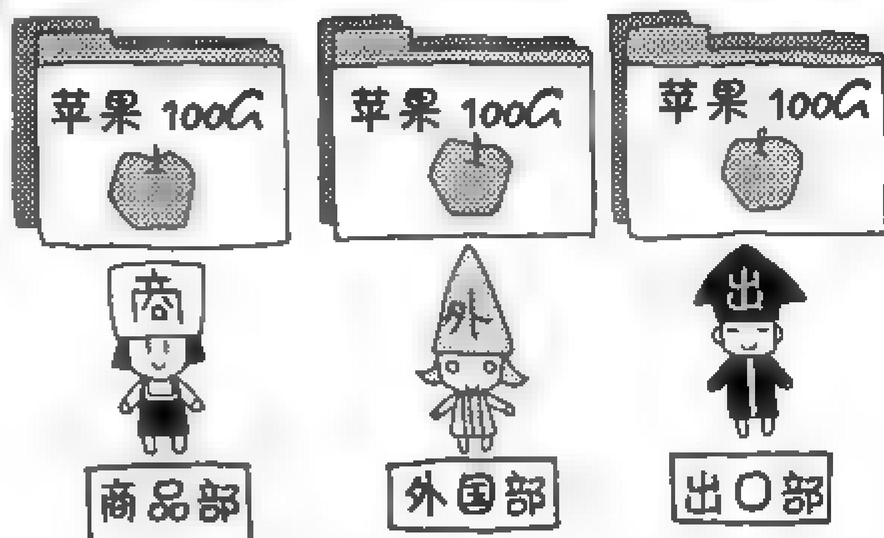
是由每个部门制作
文档来管理的。

商品
顾客
销售

由每个部门各自管理呀……

嗯嗯嗯

各自制作文档，那各部门的数据也不是重复了吗？

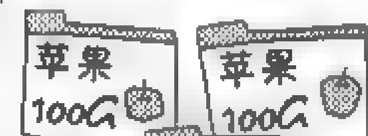


嗯

嗯

G是编码王国使用的货币单位。

是那样的！



各部门都有各自的数据……

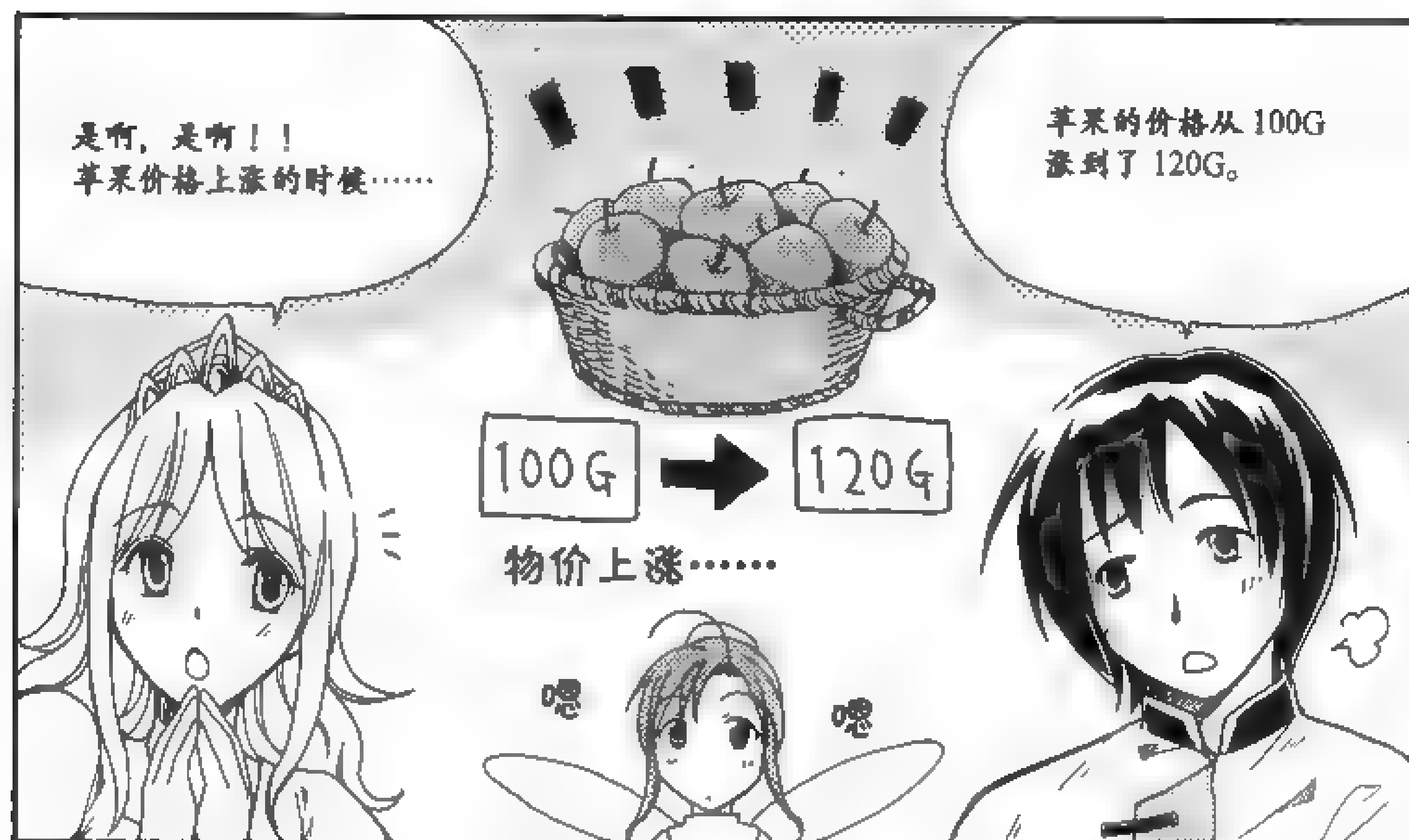
虽然小奈说，

这个原就很有效率，

但是……

有时也会出现麻烦！

说起来，前段时间就出过大麻烦呢！


是啊，是啊！！
苹果价格上涨的时候……
苹果的价格从100G涨到了120G。
100G
120G
物价上涨……
嗯
嗯

虽然通知了各部门把价格改为120G……
请把价格涨到120G！
商
外
出
但是呢？

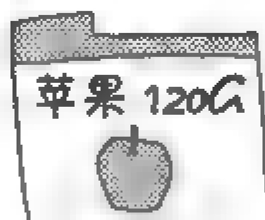
有些部门会忘记修改！！
ガーン…
不知道呀，说过吗……
外
睡着了。
外国部
苹果
100G 原封不动

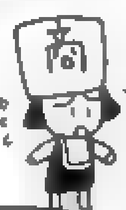
这还不算完呢……

想起来后怕

有的部门还错误地把价格改成了 300G！

大混乱!!



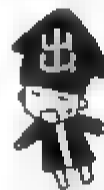


啊？？

商品部

好奇怪啊？

外国部



为什么错了？

出口部

各部门的数据出现了矛盾。

唉——

哎呀

就是这样！

在城里跑一圈更正错误——真是折腾！

特别是凯恩！

哈哈……

苹果的价格错了~

请更正价格 苹果 170G

唉~

忙乱的人

虽然父亲总是说"可以开始'编码王国水果采摘之旅'了吧！"
但是我感觉还为时尚早……
父亲很满足的样子
欢迎来到水果王国
用新系统开创新的事业吧！！
哈哈哈

虽然启动了新的事业，但是现在系统使用的系统似乎不能用啊……
变得乱七八糟的……
开始新事业的话，就必须重新制作各部门的文档。

所以，工作一点都没减少！
别着急

……是啊，一个一个输入、确认，太浪费时间了，工作量可是不少呢。
忙来忙去的，光整理数据了……
一大摞

能实现大家数据共享目的的就是数据库了！

使用它就可以舍弃无用的数据了。

嘣嘣嘣

真的吗！？

你是说可以建立一个比现在更有效的系统！？

怎么样，值得学习吧？

嗯！

我不太懂

但是……能不能拜托你帮我做啊……

……

啊啊

我没有真正的身体，所以无法使用现实世界的电脑哟！

很遗憾

原来如此

哦，是从书里出来的原因吧…

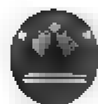
我会好好教你们的，加油噢！！

有点担心……

但是为了我和我的国家，

我就试试看！

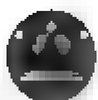
哇哦，公主！



编码王国的现状

编码王国运行着一个管理数据的系统。但是，现在的系统存在很多问题。到底是怎么回事呢？让我们来深入地分析一下编码王国的现状。

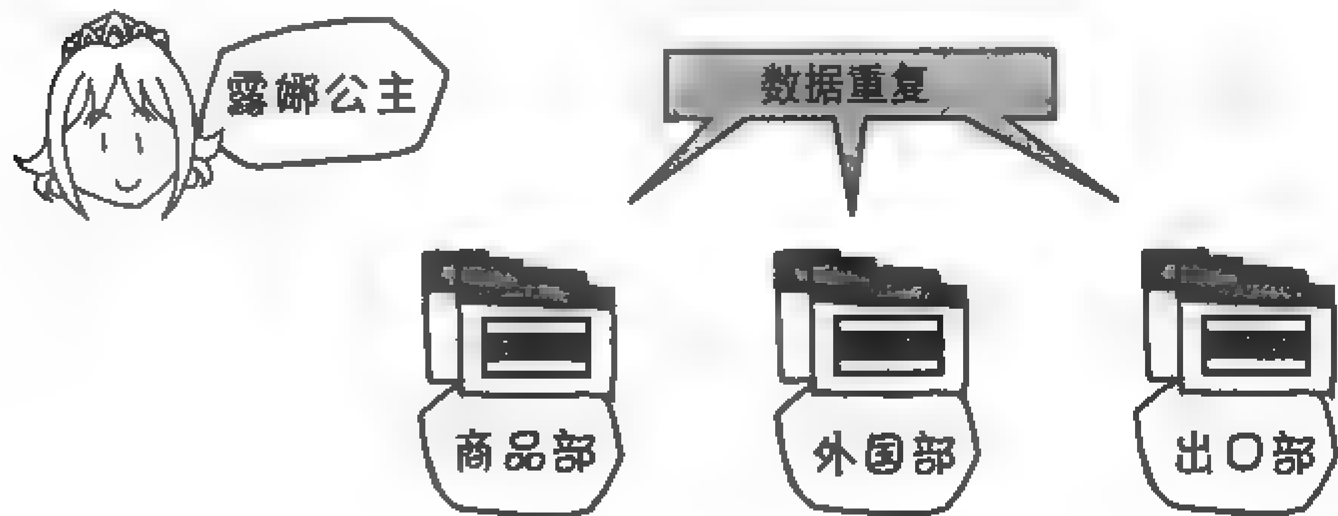
王国现在有商品部、外国部和出口部三个部门。商品部负责管理国内生产的水果。外国部负责管理与之有贸易往来的国家。出口部负责管理水果的出口。



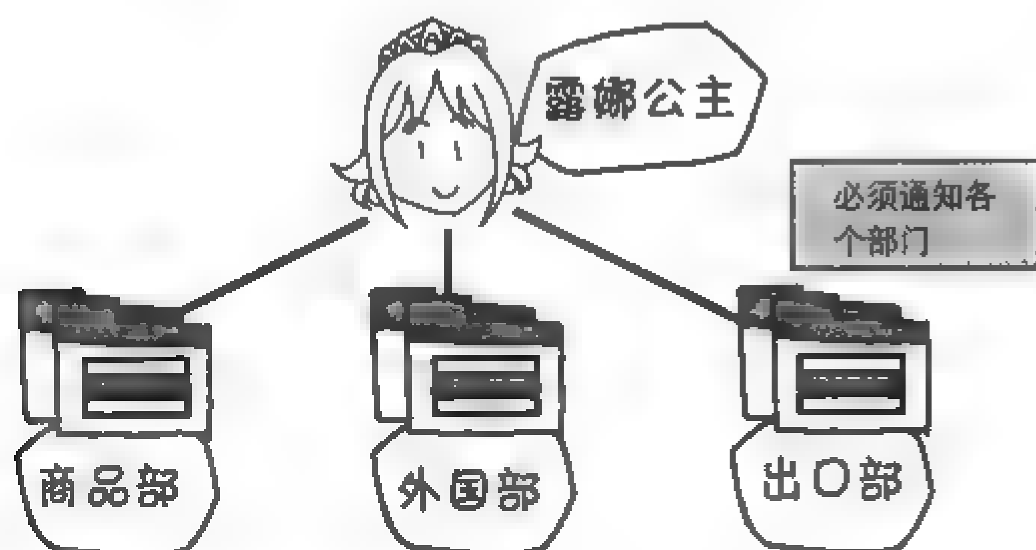
数据重复

露娜公主好像对现在的系统非常不满意。为什么不满呢？

目前，各个部门各自管理着自己一方的数据。无论是商品部还是出口部都制作自己的文档管理水果数据。这种状况下，首先是各部门的数据重复，造成了浪费，另外各部门输入数据要花费很多时间，硬盘和用来确认的纸张浪费不少。某个部门的数据自然也不能用于其他部门。



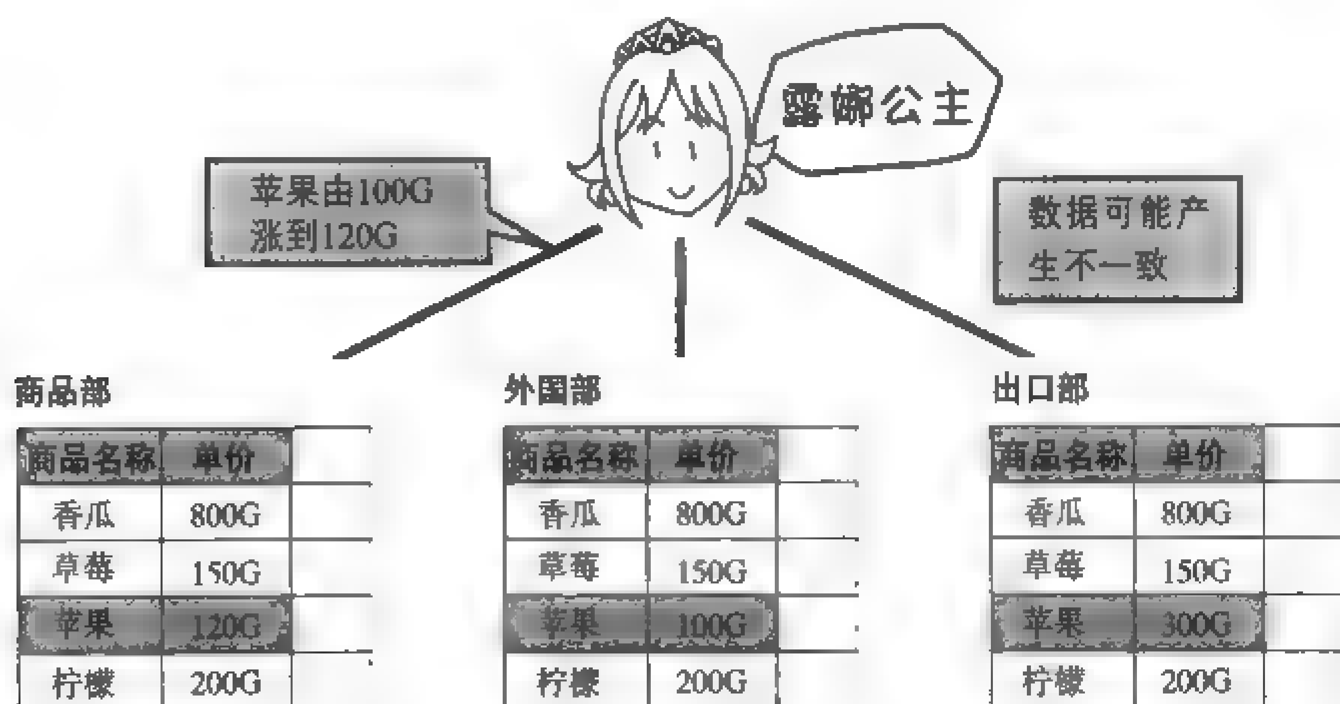
这种情况不仅造成了浪费，还会产生其他的麻烦事。例如，苹果的价格发生了变化。这时，露娜公主必须通知所有部门更改苹果的价格。这样做非常不方便。



数据有可能出现矛盾

单纯地“通知各部门苹果的价格变动了”也是件很麻烦的事情。

露娜公主正确地通知三个部门价格变更了。但是，外国部有可能忘了更改价格。另外，出口部可能把价格改成了300G。错误发生后，各部门的数据就出现不一致的情况了。系统的信息与现实世界的实际情况对应不起来了。



商品部

商品名称	单价
香瓜	800G
草莓	150G
苹果	120G
柠檬	200G

外国部

商品名称	单价
香瓜	800G
草莓	150G
苹果	100G
柠檬	200G

出口部

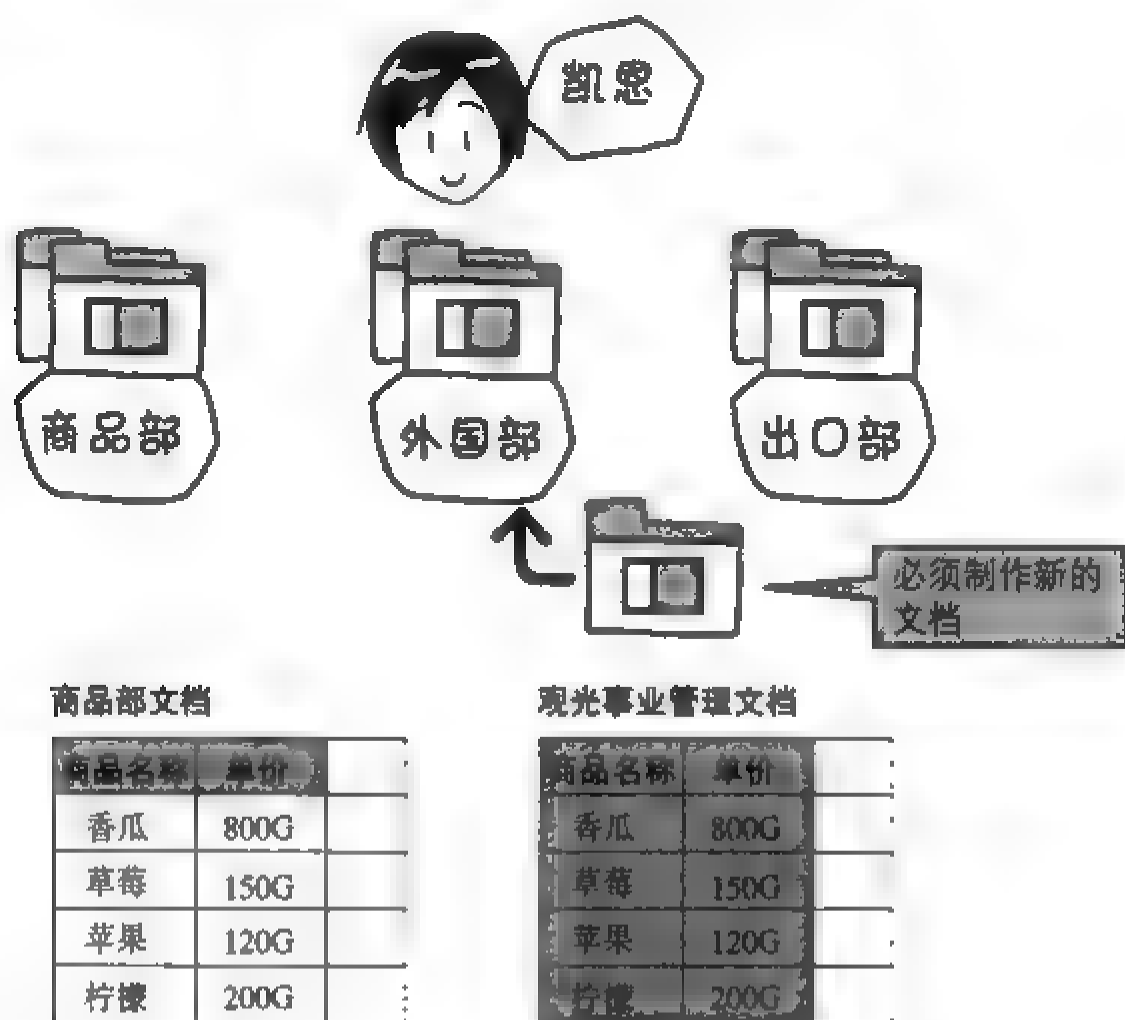
商品名称	单价
香瓜	800G
草莓	150G
苹果	300G
柠檬	200G

这样，目前运作的系统可能会出现数据矛盾，既不方便，又令人担心。

难以应对新的变化

目前的系统不仅容易出现矛盾，而且很难开展新的业务。例如，编码王国的观光业。以水果为招牌的观光业，如果能利用目前使用的水果数据的话，就可以省去输入数据的时间，这样不就很方便吗？

但是，目前系统中正在使用的数据不见得能够原封不动地被使用。水果数据由各部门分别进行管理。为了管理新的观光事业，必须另行制作观光事业用的文档。



商品部文档

商品名称	单价
香瓜	800G
草莓	150G
苹果	120G
柠檬	200G

观光事业管理文档

商品名称	单价
香瓜	800G
草莓	150G
苹果	120G
柠檬	200G

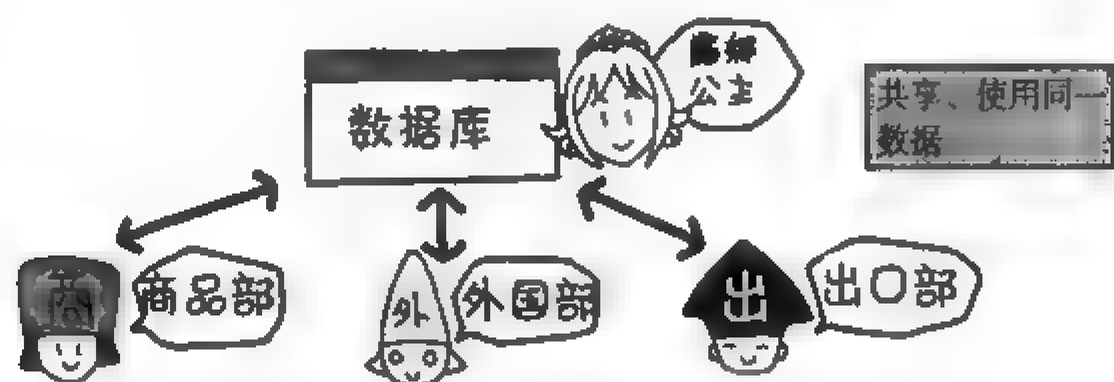
这样在开始新的业务时，又会增加更多的重复数据。

因此，可以说现行的系统并非是一个高效的系统，而是一个难以开展新业务、难以应对新环境的系统。

通过引入数据库加以解决

那么，这些不便是怎么发生的呢？归根结底这些问题都是由于各自管理数据造成的。为了更加高效地处理数据，仅仅单纯地管理数据是不够的。

那么，怎么做才更好呢？是的，对整个王国的数据进行一元化管理就可以了。这就是小 T 教给我们的“数据库”。



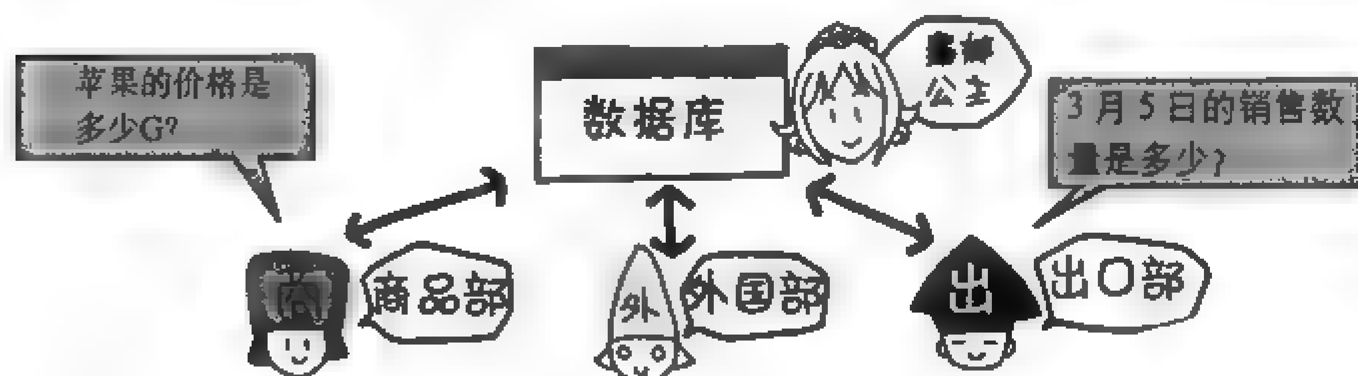
对数据库进行一元化管理，共享数据。这样，各部门都可以查询并使用数据，从而建立起了高效有序的系统。

这样既能够防止发生数据矛盾，又没有重复的数据，可以很容易地引入新的系统。

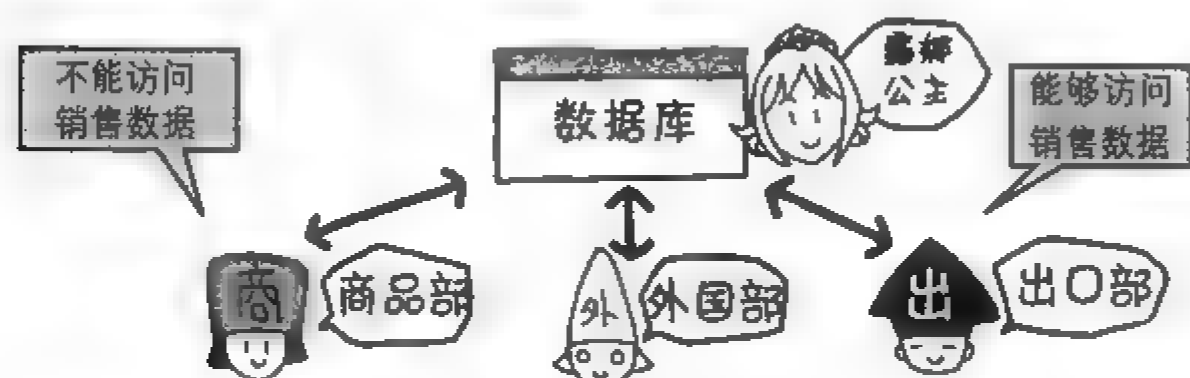
灵活运用数据库

但是，为了引入并使用数据库，需要弄清楚很多问题。

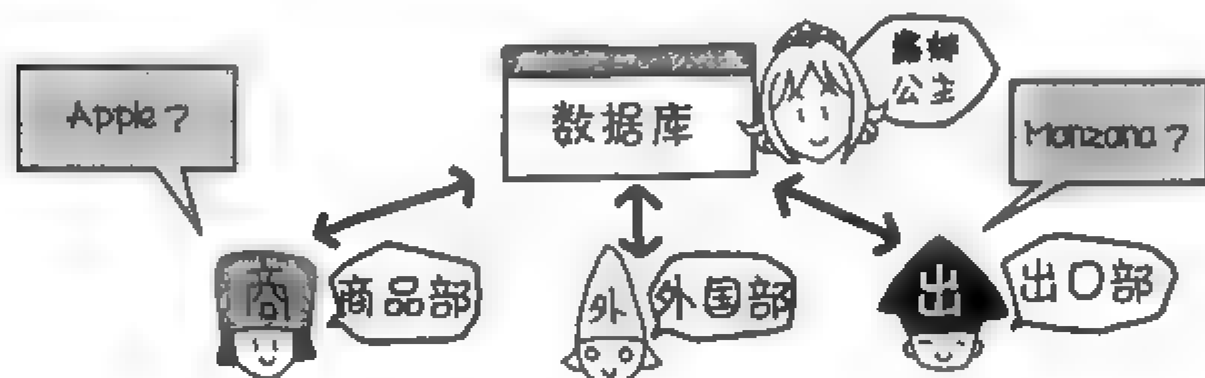
首先，各部门应怎样录入和提取数据。很多人使用一个数据库时，简单地录入、提取数据变得非常必要。我们必须使用任何人都能够理解的方法处理数据库。



另外，新的系统由于可以供多人使用，因此重要数据有被盗或被篡改的危险。例如，薪酬数据等就是只能由一部分人阅读的机密数据，销售数据只能由出口部的人更新……因此必须确保数据的绝对安全。



还有一个更难的问题，就是数据库有可能同时由多人使用。例如，外国部和出口部想要同时更改水果的名称。外国部要把苹果改成Apple(英语)，出口部要把苹果改成Manzana(西班牙语)。这时，商品名称会是什么样呢？在多人使用的数据库中，必须能很好地解决此类问题。



另外，还要注意不能丢失数据。系统有可能死机，硬盘有可能发生故障……数据损坏的危险时有发生。因此，从这些故障中恢复数据的工作就变得非常必要了。

由于数据库要处理大量的数据，因而必须具备能够进行高速检索的功能。新的系统必须是能够解决这些问题的系统。

大家是不是想尽快与露娜公主、凯恩一起学习数据库的知识来解决问题了？

小结

- 文件·应用的管理方式，数据会产生矛盾。
- 文件·应用的管理方式，数据会出现重复。
- 通过数据库可以共享数据。
- 通过引入数据库，能够防止数据的不一致和重复。
- 数据库为了实现多人共同使用数据，必须具备多种功能。

管理数据库的DBMS

我们即将学习的数据库是由一个叫做DBMS(Database Management System)的软件来管理的。

DBMS拥有多种功能，例如从数据库中提取数据的功能，防止数据不一致的功能等。另外还具备在大量的数据中进行高速检索的功能。

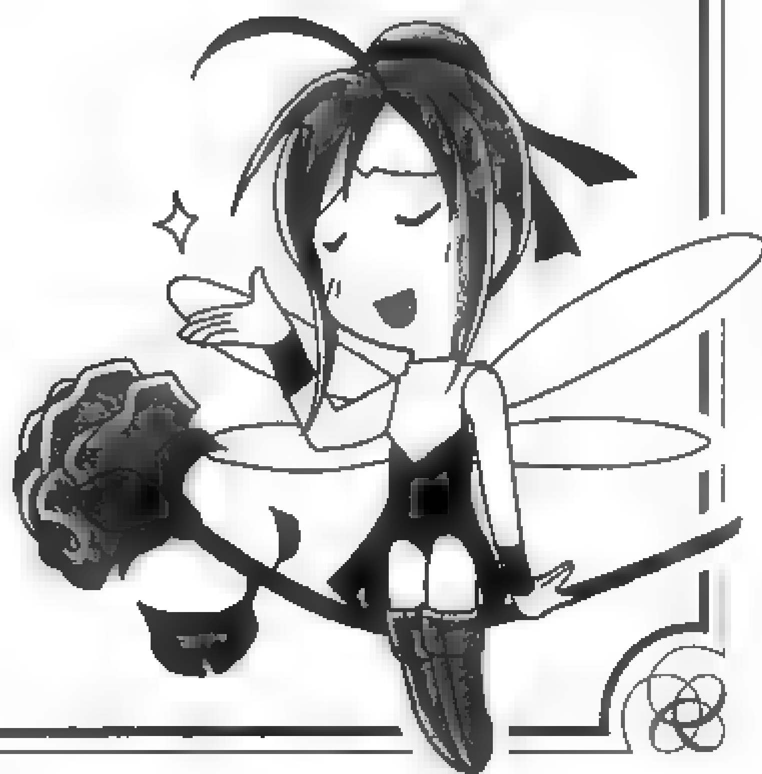
实现多人共同使用数据库，DBMS功不可没。DBMS具有让众多用户同时正确使用数据库的控制功能。

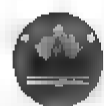
另外，DBMS还具有保护数据库安全的功能。在发生故障时，它能够使数据库正常工作。

DBMS连接着数据库与用户，使人们能够正确地使用数据库。接下来让我们一起学习数据库的知识和DBMS的功能吧。

第 2 章

关系数据库是什么





了解数据库的术语

哗

露娜，亲爱的！

美丽的

公主

……

嗯？

怎么了，亲爱的～

可是，拉来内斯是邻国的王子，

是不能怠慢的啊！

我也知道～～～

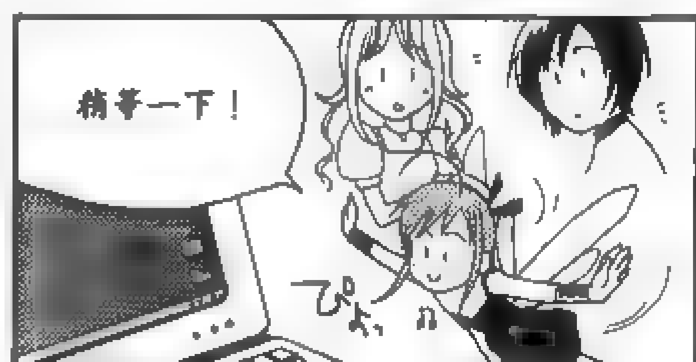
真够乱的…

哇
哇，什么时候来的！
刚才凯恩打开书的时候！
哈喽～♡

还不够乱吗？
真是吓了一跳……
好了。
拉来内斯已经回去了。

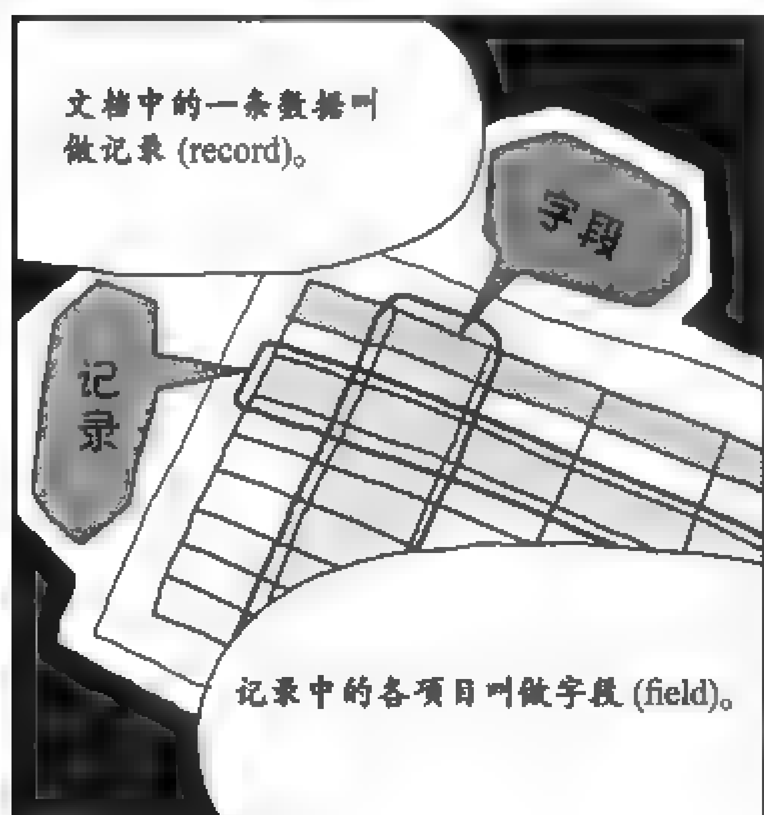
我们开始设计数据库吧！
Stop
等……等一下
Stop!!

不懂基本知识是设计不出数据库的哟！
基础知识非常重要！！
嗯……
哦——
加油啊
首先，要先了解数据库的基本术语！


稍等一下！

！！！

商品编码	商品名称	单价	备注
101	香瓜	800G	种子
102	草莓	150G	
103	苹果	120G	
104	柠檬	200G	
201	栗子	100G	
202	柿子	160G	
	桃子		



任何一条记录所对应的同一字段值都是同种类型的。

商品编码	商品名称	单价	备注
101	香瓜	800G	带籽儿
102	草莓	150G	
103	苹果	120G	
104	柠檬	200G	酸味
201	栗子	100G	带皮
202	柿子	160G	
301	桃子	130G	
302	猕猴桃	200G	贵重品

记录

字段

原来如此！

例如，商品编码是三位数，

商品名称是十位以下的字符。

商品编码
1·0·1
1·0·2
1·0·3
1·0·4
2·0·1
2·0·2

商品名称
香瓜
草莓
苹果
柠檬
栗子
柿子

商品编码
101
102
103
104
201
202
301
302

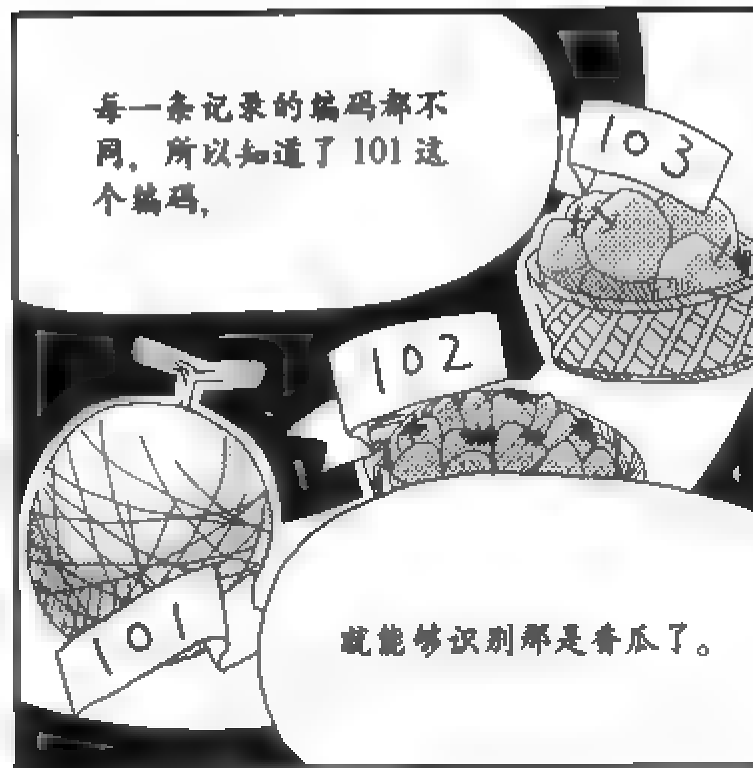
这里

接下来我们详细介绍一下商品编码吧！

记录

字段

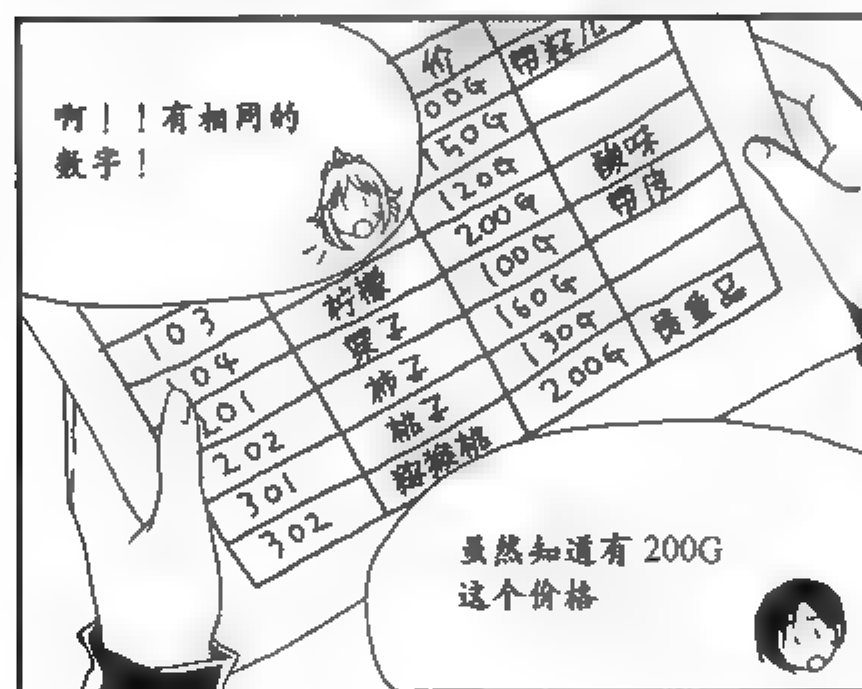
商品编码是
不重复的。
这样啊！


每一条记录的编码都不
同，所以知道了101这
个编码，
就能够识别那是香瓜了。
103
102
101

那是当然的
这个懂
了吧！
但是，这个
…

单价
800G
150G
120G
200G
如果是单价的话，会怎么
样呢？

单价？


啊！！有相同的
数字！
虽然知道有200G
这个价格

但它是柠檬还是猕
猴桃就不知道了。
200G
200G

用商品编码可以识别数据，用单价就不能了。

是啊，是啊！

在数据库的世界里像商品编码这样的不能重复的值，

叫做唯一 (unique)

唯一？

商品编码是唯一的

爸爸也经常说这个词……

嘿，嘿！

哈哈哈哈哈哈哈

编码王是唯一的

就是只有一个！

One!!

Only——

也有那个意思。

接下来我们来看备注，

备注？

备注不就是备注嘛？

写着特征

人物	备注
露娜	金发 活泼
凯恩	黑发 木讷

发呆

让我们从数据库的角度来看吧！

备注与其他字段不同，可以不输入数据。

	备注
G	带籽儿
0G	
20G	酸味
200G	带皮

是这样啊……

商品编码	商品名称	单价	备注
101	香瓜	800 G	带籽儿
102	草莓	150 G	
103	苹果	120 G	
104	柠檬	200 G	酸味
201	栗子	100 G	带皮
202	柿子	160 G	
301	桃子	130 G	
302	猕猴桃	200 G	

但是……就这么继续使用现在的商品账簿文档的话，
嘿
哗
是不能解决诸多问题的……
哗

是啊！！
所以，好想做个数据库啊！

所以，赶快教我们吧！！
还不行，还不行。
No！

例如，
请
看
数据
数据
数据
数据
数据
数据
数据
数据之间存在着像树一样的层级关系的，
叫做层次数据模型(Hierarchical data model)。
哇！
又变出什么了！！

数据库有很多种哟！
跟水果似的
是这样啊？

数据之间存在着像网一样的关系的，叫做网状数据模型(network data model)。

碎

层次模型

好镇定啊 凯恩

又变出东西了

那么该用哪一个呢？

都不用！！

No!

No!

碎

碎

碎

啊

其实还有比这两个更好用的一种数据库。

这回吓到了吧

没有啦

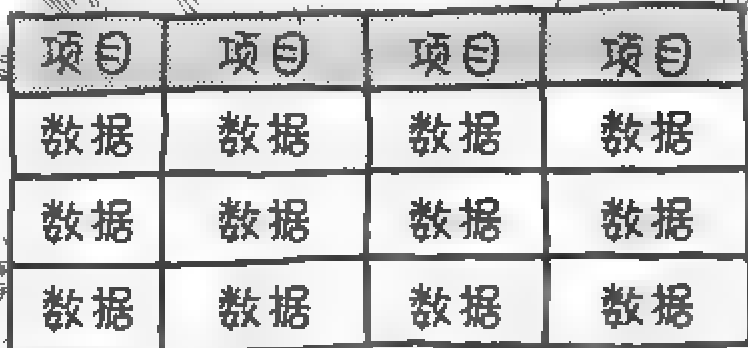
呵呵

那就是··

关系数据模型 (relational data model) ！

关系？

使用表格的关系数据库



关系数据模型以二元表格(table)为基础

砰 砰

看一下

咦

又变出什么了……

砰，砰

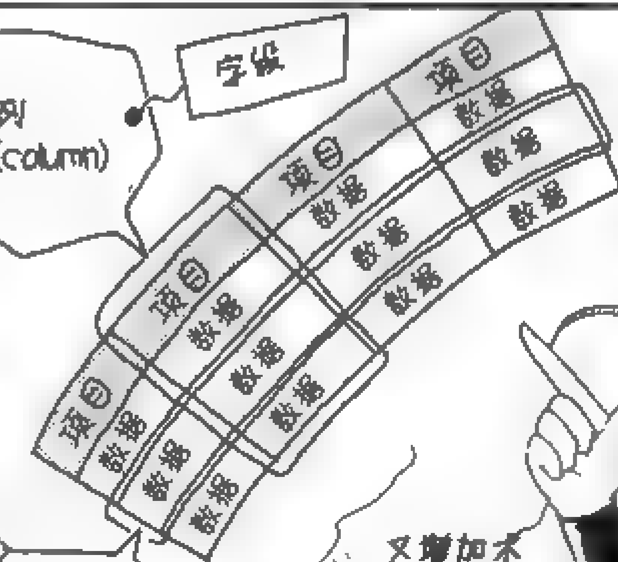
对吧

商品等数据很容易做成表格。

在关系数据模型中，我们称表格为关系（relation）。

咦……

字段



砰

砰

一条数据也就是记录我们称之为行(row),

各数据项目也就是字段，我们称之为列(column)。

又增加术语了！！

记录

在数据库中如果赋予字段重要的角色，
这时，这个字段就称作键(Key)。
Key
重要的角色？
是的，例如
之前我们看到的商品账簿文档中的商品编码，
像商品编码这样具有识别数据重要功能的字段
我们称为主键(Primary key)。
主关键词
商品编码
101
102
103
104
201
202
301
302
好多术语啊··
用表格处理数据的话，似乎很容易理解啊。
意义重大
是的！！这就是关系数据模型的好处。
不熟悉数据库的人也可以处理数据！

而且关系数据模型可以通过基于数学概念的运算来，

处理数据……

哦……

数学？

果然是很难的样子……

唉——

没有那么难啦！

商品编码	商品名称	单价	备注
101	香瓜	800G	带籽儿
102	草莓	150G	
103	苹果	120G	
104	柠檬	300G	酸味
201	栗子	100G	带皮
202	柿子	160G	
301	桃子	130G	
302	猕猴桃	300G	贵重品

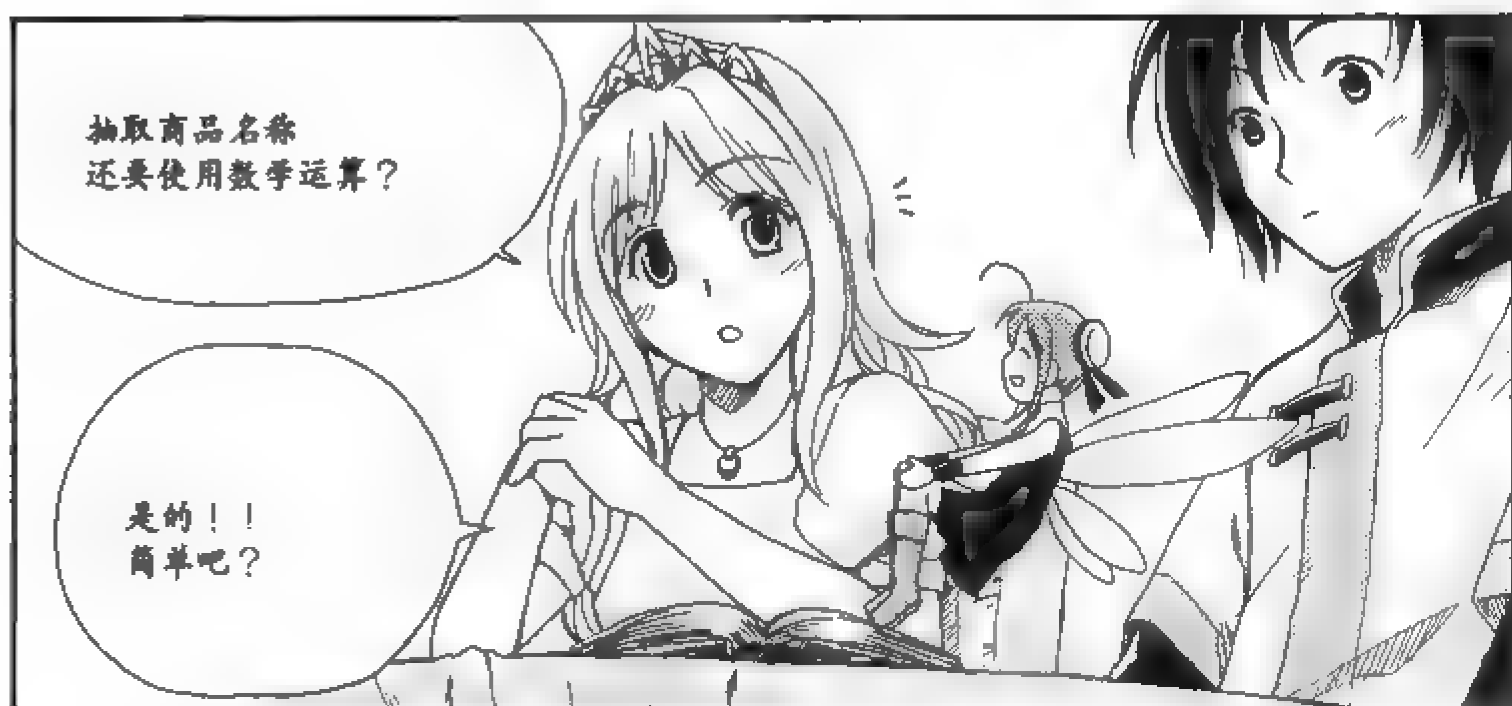
商品名称
香瓜
草莓
苹果
柠檬
栗子
柿子
桃子
猕猴桃

例如，我们再回到商品表，

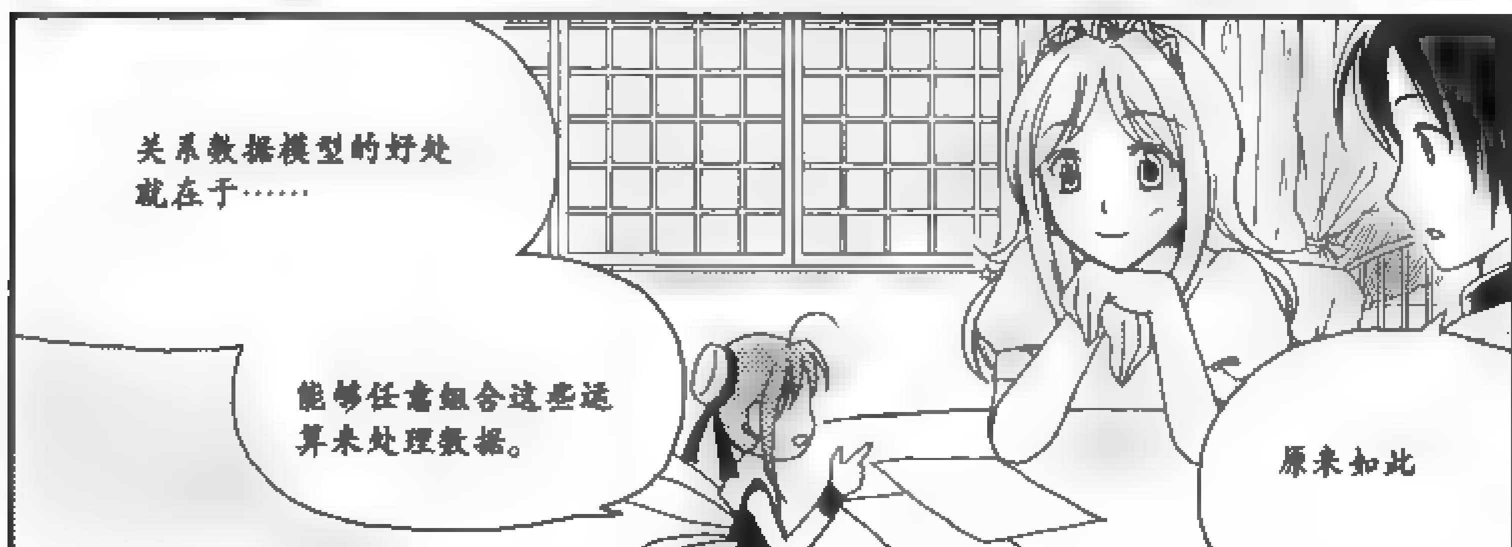
把商品名称抽出来了。

像这样抽取列的运算我们叫做……

投影（projection）。


抽取商品名称
还要使用数学运算？
是的！！
简单吧？

还有很多其他运算。
有8种！！
哇！
这么多啊！
并
差
笛卡儿积
除
投影
交
选择
连接


关系数据模型的好处
就在于……
能够任意组合这些运
算来处理数据。
原来如此

那么，我们一起来做
关系数据库吧！
嗯！！
ちょん

拉米内斯！

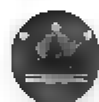
谁…？

拉米内斯殿下刚才
出去了…
啊……
那个！

抱歉，我们捉
迷藏呢！
ほほほほほほ

拉米内斯
噗通

哈哈哈
那个家伙……
到底有多少个
女孩呀？！
哇—

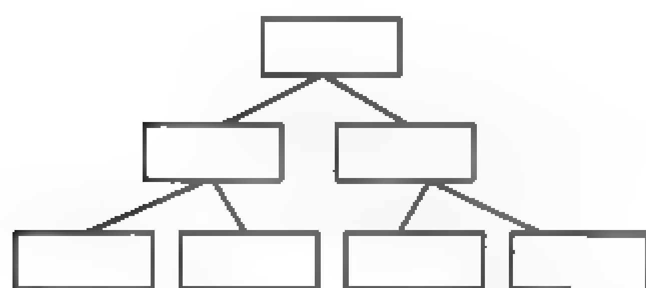


数据模型的种类

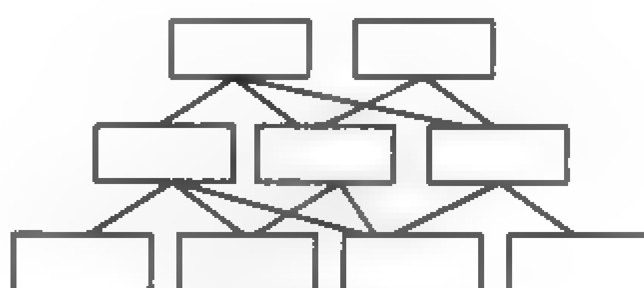
数据库有很多种。在管理数据时应该使用哪种形式呢?

数据之间有着怎样的关系，又该如何处理这些数据……将这些问题模型化，就是我们说的数据模型 (data model)。最具代表性的有 3 种数据模型。

首先是小 T 最先提到的层次数据模型 (Hierarchical data model)。在层次数据模型中每个子数据都有一个母数据。其次还有网状数据模型 (network data model)。网壮数据模型与层次数据模型不同，每个子数据可以有多个母数据。

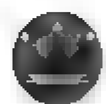


层次数据模型



网状数据模型

这些模型用指针 (Pointer) 连接数据表示它们的关系。指针就是表示数据在硬盘上的存储位置的架构。由于这些模型由指针连接，就必须在知道数据的物理位置和构造的情况下去处理数据。因此，灵活高速地检索数据非常困难，这使得以上两种模型都难以成为一种谁都可以操作的数据库。



关系数据库

第三个登场的是关系数据模型 (relational data model)。作为当今主流数据库的关系数据库，就是以关系数据模型为基础的。关系数据库使用的是更易于理解的表格，有了它们就可以处理数据了。

关系数据库可以基于数学运算进行数据操作。通过反地严密定义的运算，能够抽取数据生成表格。

并(union)

那么，在关系数据库中用什么方法抽取数据呢？我们来看下面的商品表 1 和商品表 2。

商品表1

商品名称	单价
香瓜	800G
草莓	150G
苹果	120G
柠檬	200G

商品表2

商品名称	单价
香瓜	800G
草莓	150G
栗子	200G
柿子	350G

运行并运算 (union) 后，能够抽取包含在商品表 1 和商品表 2 中的所有商品。

并

商品名称	单价
香瓜	800G
草莓	150G
苹果	120G
柠檬	200G
栗子	200G
柿子	350G

并是抽取两个表格中所有行的运算，如下图所示。并运算能够抽取存在商品表 1 中的行和存在商品表 2 中的行。

差(difference)

能够调取商品表 1 和商品表 2 中其中一张表格独有的商品，这种运算叫做差 (difference)。差是抽取其中一张表格独有行的运算。以不同的表格为基准会获得不同的运算结果。

差

商品名称	单价
苹果	120G
柠檬	200G

商品名称	单价
栗子	200G
柿子	350G

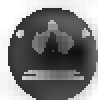
交(intersection)

能够调取商品表 1 和商品表 2 两张表格中都存在的商品，这种运算叫做交 (intersection)。

交

商品名称	单价
香瓜	800G
草莓	150G

它可以调取“既在商品表1中存在也在商品表2中存在的行”。

 笛卡儿积(Cartesian product)

笛卡儿积(Cartesian product)是一种可以将两个表格中所有行排列组合的方法。例如下图中的商品表和出口国表。

商品表

商品编码	商品名称	单价
101	香瓜	800G
102	草莓	150G
103	苹果	120G

3行

出口国表

出口国编码	出口国名称
12	米纳米王国
23	阿尔法帝国
25	理陀儿王国

3行

笛卡尔积

商品编码	商品名称	单价	出口国编码	出口国名称
101	香瓜	800G	12	米纳米王国
101	香瓜	800G	23	阿尔法帝国
101	香瓜	800G	25	理陀儿王国
102	草莓	150G	12	米纳米王国
102	草莓	150G	23	阿尔法帝国
102	草莓	150G	25	理陀儿王国
103	苹果	120G	12	米纳米王国
103	苹果	120G	23	阿尔法帝国
103	苹果	120G	25	理陀儿王国

3×3=9行

笛卡儿积将两个表格的行全部组合排列。这样就得到了“3行×3行 9行”。

投影(projection)

并、差、交和笛卡儿积称为集合运算。集合运算是高等数学中比较重要的运算。

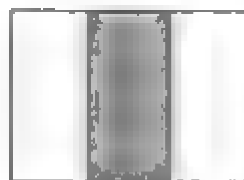
让我们来看看其他运算。下面这些运算是关系数据库特有的运算。

投影(projection)是调取表中某一列的运算。仅调取商品表中商品名称时可以使用该运算。

投影

商品名称
香瓜
草莓
苹果
柠檬

用图表示就是下面这种情况。



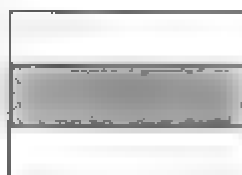
选择(selection)

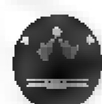
选择(selection)是调取表中某行的运算。从前页中的商品表中调取多行就可以用这个运算。

选择

商品名称	单价
香瓜	800G
草莓	150G

用图表示就是下面这种情况。





连接(join)

关系数据库中还有功能更强大的运算，就是连接运算。顾名思义，连接就是将表格“连接起来”。

例如，我们来看下面的表格。

商品表

商品编码	商品名称	单价
101	香瓜	800G
102	草莓	150G
103	苹果	120G
104	柠檬	200G

销售表

日期	商品编码	数量
11/1	102	1100
11/1	101	300
11/5	103	1700
11/8	101	500

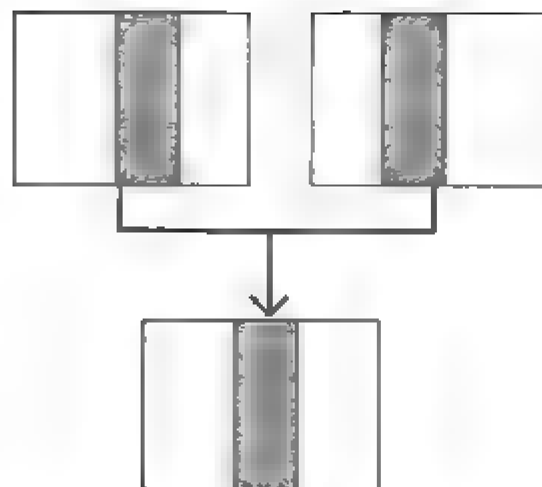
这两个表格中的“商品编码”列其实表示的是相同的项目。例如，11 月 1 日商品编码为 102 的草莓卖出了 1100 个。销售表中没有记录商品名称但记录了商品编码，所以能够知道是哪个商品。也就是说，在销售表中可以通过参照商品表中作为主键的商品编码，来分辨到底是哪个商品。此时，销售表中的商品编码就叫做外键 (foreign key)。

外键参照其他表中的主键时，通过连接，可以将两个表格粘合成如下表格。

连接

日期	商品编码	商品名称	单价	数量
11/1	102	草莓	150G	1100
11/1	101	香瓜	800G	300
11/5	103	苹果	120G	1700
11/8	101	香瓜	800G	500

这样就生成了包含日期和商品名称的销售表。自然也就能够从两个表格中调取必要的数据了。



除(division)

最后让我来看看除运算。除就是从“被除表格”中调取“除表格”中包含的所有行，然后再从中去掉“除表格”中所有行的运算。让我们来看一个实例。

销售表(被除表格)

出口国编码	出口国名称	日期
12	米纳米王国	3/5
12	米纳米王国	3/10
23	阿尔法帝国	3/5
25	理陀儿王国	3/21
30	萨藏纳王国	3/25

出口国表(除表格)

出口国编码	出口国名称
12	米纳米王国
23	阿尔法帝国

销售表除以出口国表得出如下结果。

除

日期
3/5
3/10

首先从销售表中调取含有出口国表中所有接排数据的行。然后，从中剔除掉出口国表中的行。结果我们就可以调查出口国表中出口国全部有销售记录的销售日期了。

投影、选择、连接、除等运算我们称之为关系运算。关系数据库就是通过使用集合运算和关系运算来调取数据的。

那么，接下来为了巩固我们所学的关系数据库的知识，大家试着来回答一下下面的问题吧。

关系数据库中，参照其他表格中列的关键词叫什么？

下表为记录书籍信息的表格。哪个项目可以作为主键？书籍序号为连号。书籍名称有可能存在重复的情况。

书籍序号	书籍名称	作者	出版日期	价格

下图所示的调取数据的运算叫什么运算？

出口国编码	出口国名称
12	米纳米王国
23	阿尔法帝国
25	理陀儿王国
32	萨藏纳王国

出口国编码	出口国名称
25	理陀儿王国

下图所示的调取数据字号的运算叫什么运算？

出口国编码	出口国名称
12	米纳米王国
23	阿尔法帝国
25	理陀儿王国
32	萨藏纳王国

出口国编码	出口国名称
15	帕罗努国
22	托康塔王国
31	塔哈鲁王国
33	马里昂国

出口国编码	出口国名称
12	米纳米王国
15	帕罗努国
22	托康塔王国
23	阿尔法帝国
25	理陀儿王国
31	塔哈鲁王国
32	萨藏纳王国
33	马里昂国

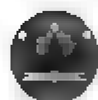
下图所示的调取数据的运算叫什么运算？

出口国编码	出口国名称
12	米纳米王国
23	阿尔法帝国
25	理陀儿王国
32	萨藏纳王国

出口国编码	日期
12	3/1
23	3/1
12	3/3
32	3/5
12	3/6
25	3/10

出口国编码	日期	出口国名称
12	3/1	米纳米王国
23	3/1	阿尔法帝国
12	3/3	米纳米王国
32	3/5	萨藏纳王国
12	3/6	米纳米王国
25	3/10	理陀儿王国

怎么样，大概了解关系数据库的基础知识了吧？



关系数据库的普及

正如上文所述，关系数据库可以通过明确的运算调取数据。调取的结果仍然是以表格的形式呈现。

组合使用我们所介绍的运算，就能够按照各种需要调取数据。可以调查商品的名称、价格，也可以生成销售统计数据。

由于可以如此简便且灵活地处理数据，关系数据库得到广泛普及就不足为怪了。

小结

- 一条数据称作记录，各个项目称作字段。
- 能够确定数据的列叫做主键。
- 关系数据库能够使用表格来处理数据。
- 关系数据库可以基于数学运算来处理数据。

答案

A1 外键

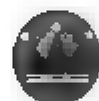
A2 书籍序号

A3 选择

A4 并

A5 连接

第 3 章
设计数据库

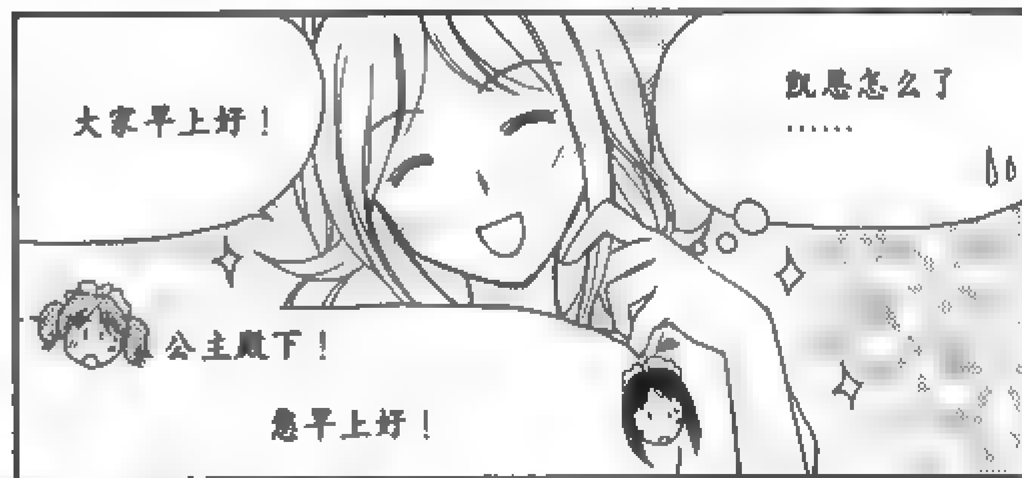


使用E-R模型来分析

凯恩，
你在哪儿？

…是啊，
凯恩先生不知道怎么了，
!!
一个人在那自言自语……

数据库是什么东西……
嗯！


大家早上好！
凯恩怎么了……
公主殿下！
您早上好！

……
是这样吗？
原来如此！

凯恩！！

じっ
嗨——
早上好，小T！
啊！
露娜公主！
你小心点，凯思！
早上好，露娜！
啊？

首先，如果能够
掌握编码王国的
现状的话，
首先
掌握
现状！！
就可以很顺利地
设计数据库了。

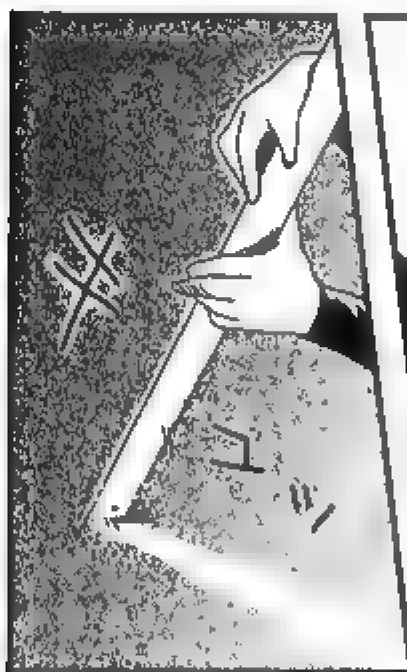
先将现在的出口管
理现状制作成模型，
这样便于分析……
要说很久，
先请坐吧♪
谢谢
咔嗒
又能学习新东西了。

那么，
准备好了吗？
这个分析中使用
的模型叫做…
E-R
"E-R 模型"！

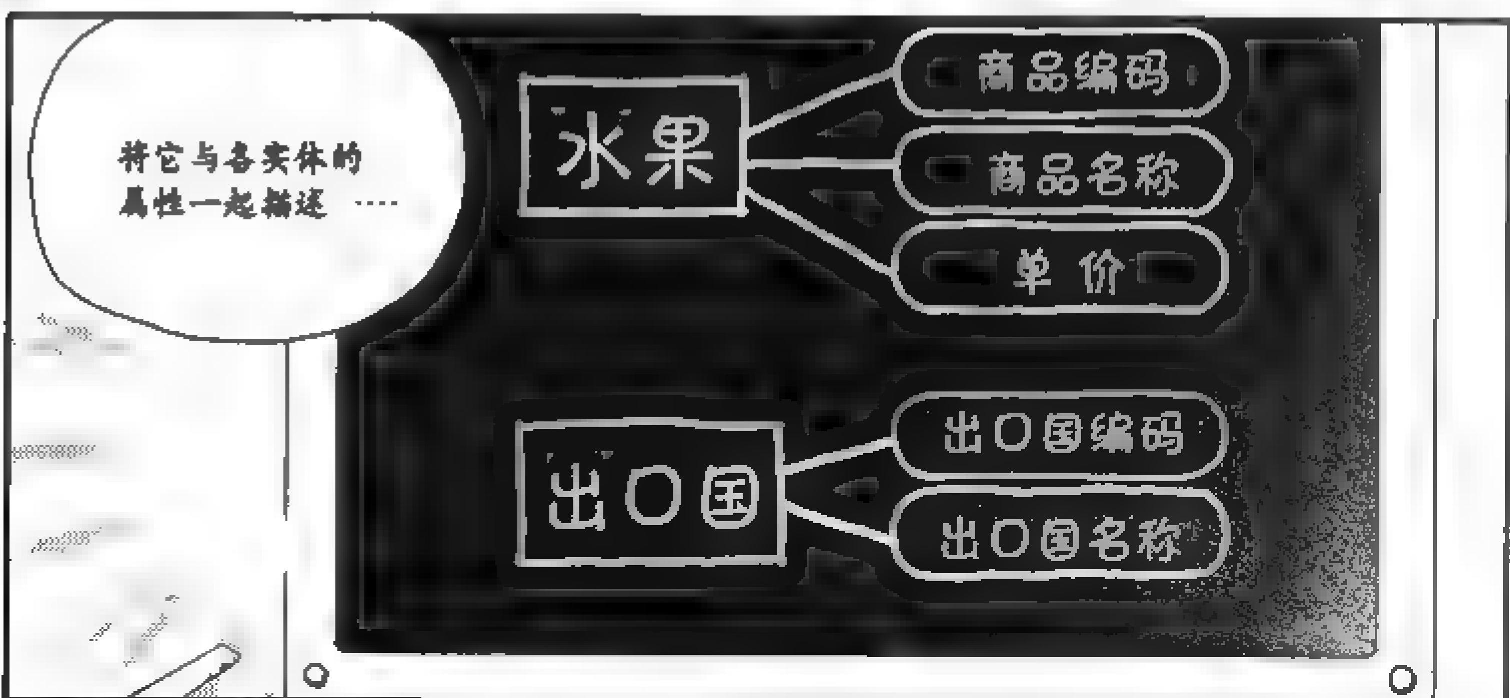
E 就是 Entity（实体），
R 就是 Relationship（关系）。
E=实体
R=关系
"E-R"……

实体和关系……
是
有点不太明白……

用"实体"和"关系"
这两个概念来考虑
现实世界的模型就是
E-R 模型了。
具体来说……

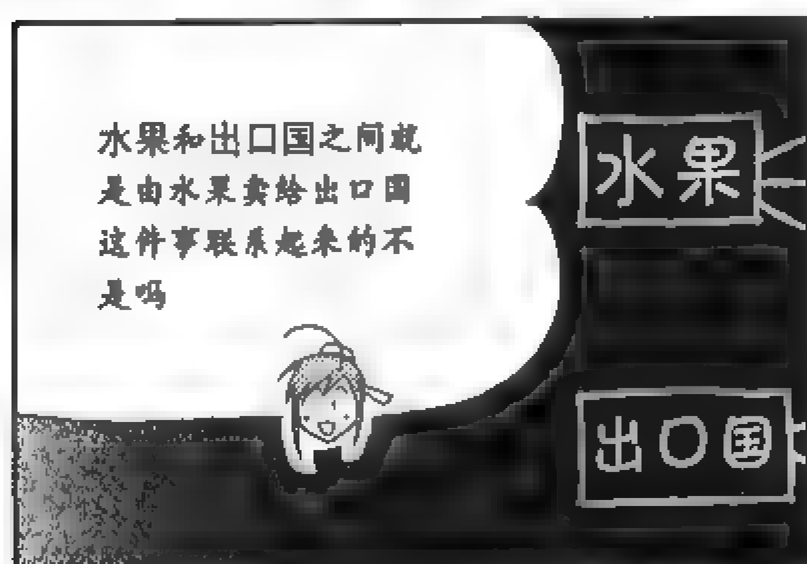


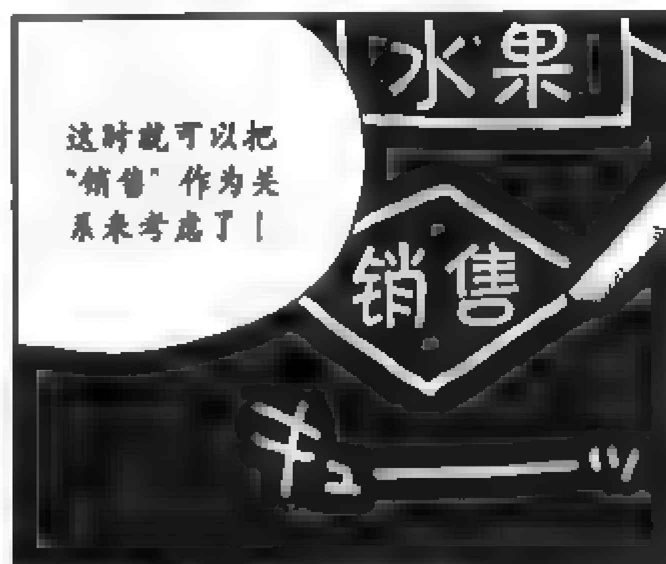
所谓实体就是能够
认知的现实世界的
事物，
例如将水果出口至其他国
家这件事，就可以将水果
和出口国作为实体来考虑。

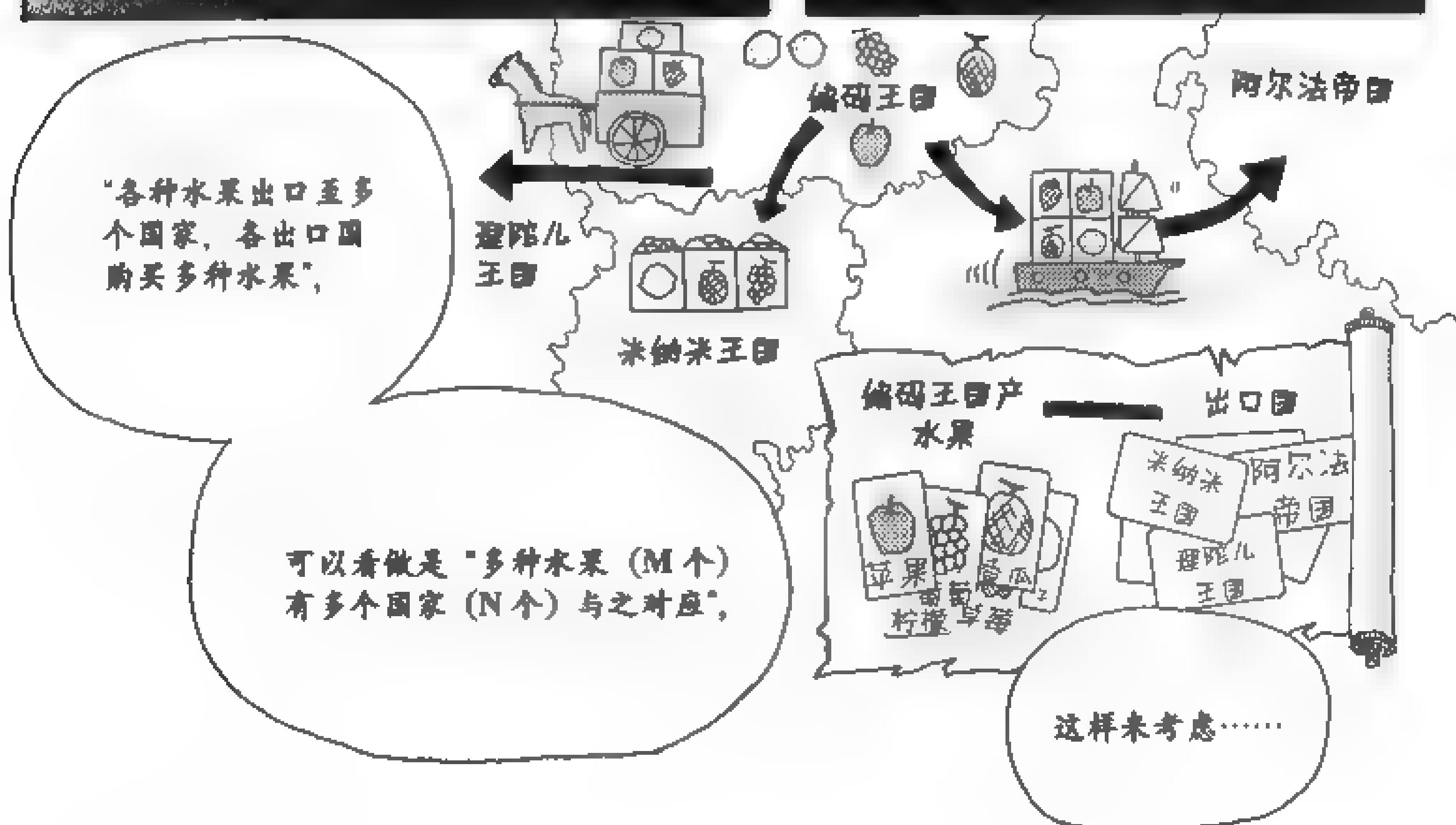

将它与各实体的
属性一起描述……
水果
商品编码
商品名称
单价
出口国
出口国编码
出口国名称

就是分析水果和出口国了。
那"关系"呢？
关系就是表示实体之间有着怎样的关连。

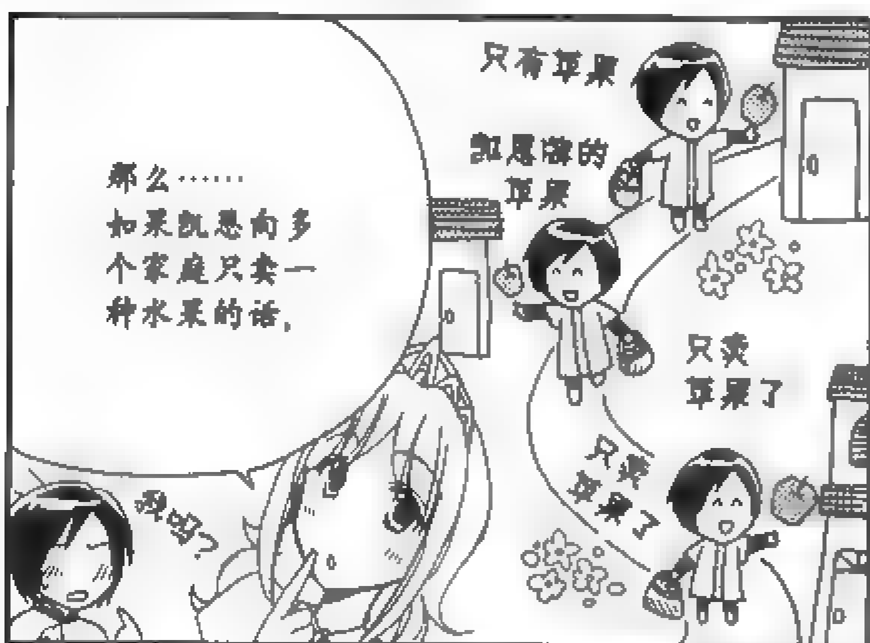
例如……


水果和出口国之间就是由水果卖给出口国这件事联系起来的不是吗
水果
出口国

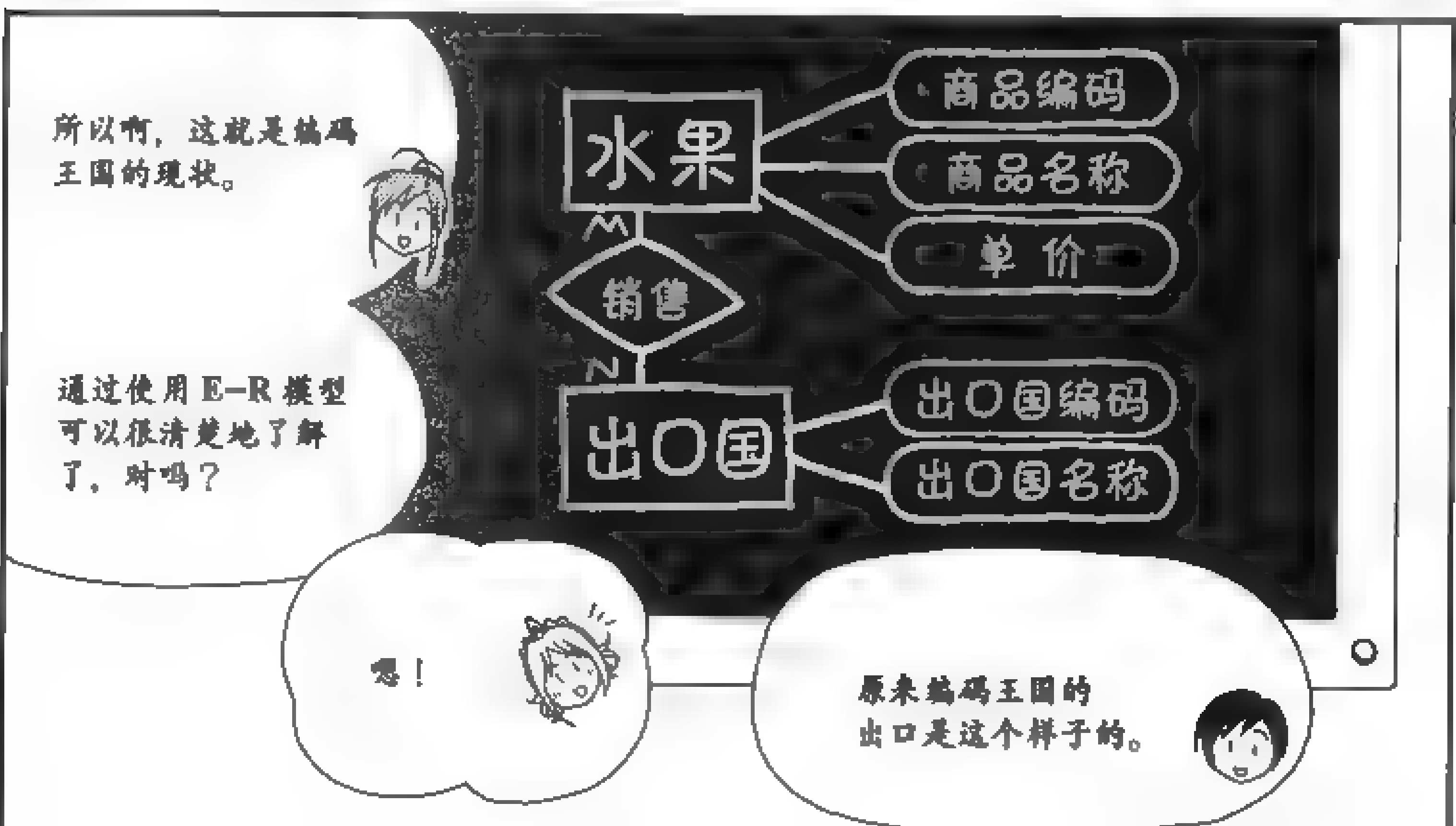

这时就可以把"销售"作为关系来考虑了！
水果
销售

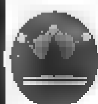

"各种水果出口至多个国家，各出口国购买多种水果"，
可以看做是"多种水果（M个）有多个国家（N个）与之对应"，
编码王国
阿尔法帝国
米纳米王国
编码王国产水果
出口国
苹果
柠檬
草莓
米纳米王国
阿尔法帝国
这样来考虑……

苹果
葡萄
香瓜
水果
商品编码
商品名称
单价
M
销售
N
出口国
出口国编码
出口国名称
这种状态称为“多对多”。
也就是指E-R模型中实体之间的关系个数。


那么……
如果凯恩向多个家庭只卖一种水果的话，
只有苹果
凯恩牌的苹果
只卖苹果了
只卖苹果了
我吗？

就是“一对多”？
凯恩牌的苹果
水果
销售
托马斯家
乔治家
马绍尔家
出口国
是的！
对了


所以啊，这就是编码王国的现状。
通过使用E-R模型可以很清楚地了解了，对吗？
水果
商品编码
商品名称
单价
M
销售
N
出口国
出口国编码
出口国名称
嗯！
原来编码王国的出口是这个样子的。



规范化表格

虽然很想做数据库，但是好难呀！

是啊！首先分析现状就非常重要。

那么掌握了编码王国的现状后……

欧耶！！

就可以考虑设计真正的数据库了！

公主？

钻进去……

找到了！找到了！

销售报表

1101 米纳米王国 日期：3月5日

101 香瓜 @800G × 1100 = 880000G

102 草莓 @150G × 300 = 45000G

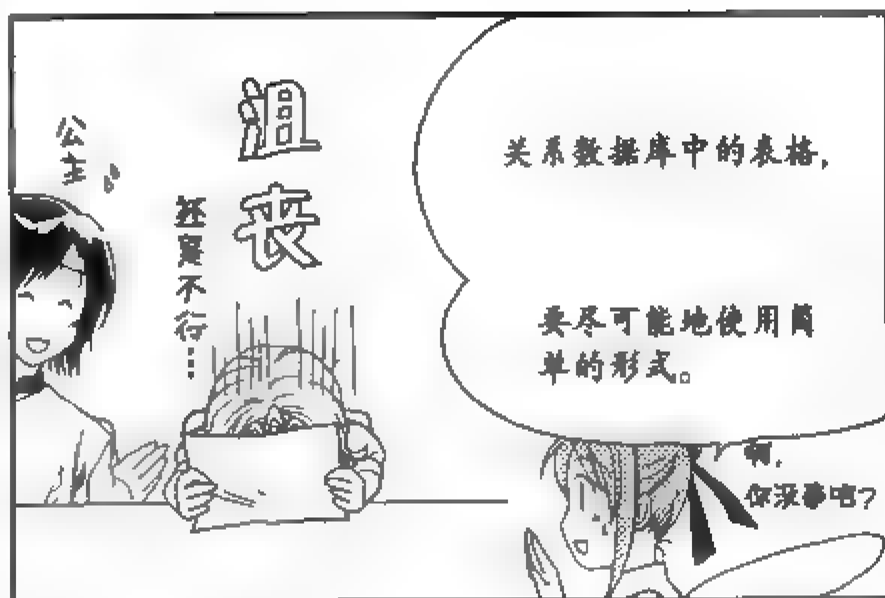
编码王国

合计 925000G

报表编码	日 期	出口国编码	出口国名称	商品编码	商品名称	单价	数量
1101	3月5日	12	米纳米王国	101	香瓜	800G	1100
				102	草莓	150G	300
1102	3月7日	23	阿尔法帝国	103	苹果	120G	1700
1003	3月8日	25	哩叻儿王国	104	柠檬	200G	500
1004	3月10日	12	米纳米王国	101	香瓜	800G	2500
1105	3月12日	25	哩叻儿王国	103	苹果	120G	2000
				104	柠檬	200G	700

由销售报表制成的表格

好厉害！！
把这个表格用于数据库的话就行了！
NO
很遗憾，
即使这样还不能用！


公主
沮丧
还是不行……
关系数据库中的表格，
要尽可能地使用简单的形式。
啊，
你没事吧？

这张表不容易理解吗？
看一下……
我就不明白~

报表中一行数据中
却有两行商品项目，
不是吗？
啊……

这是因为一张报表可以处理两种以上商品的缘故。
是的，是的！
销售报表
1101 米纳米王国 日期：3月5日
101 香瓜 @800G x 1100 = 880000G
102 草莓 @150G x 300 = 45000G
合计 925000G
编码王国

如果一行里只有一个数值的表格就很容易理解了。
嗯，
真是啊！
嗯嗯嗯

因此，

不能把数据就那么放在一个表格里……

ばん!!

要分成多个表格！

这样的！！

看！！

是这样啊！？

那样的话感觉好麻烦…
好不容易才弄成一张表！

可能会觉得很麻烦，

但是为了正确无误地管理数据这就非常重要了。

好麻烦呀……

用E-R模型掌握整体……

念念有词

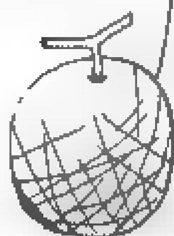
嗯……

凯恩，你干嘛呢？

啊，没什么，就是复习一下。

好喜欢自言自语呀……

例如，
"想要将香瓜的
单价上涨 20G"，



用这张表的话要在
整个表中找出所有
"香瓜"对应的行，

更改单价才行，
是吗？

但是，如果把它
分成只与"商品"
相关的表格，

商品表

香瓜	800G
草莓	150G
苹果	120G
柠檬	200G

只改这里！！

好轻松啊~

只要更改商品表中
香瓜的单价这一行
就可以了！

不会出现忘了更改某一
行，而发生数据矛盾的
情况！！

厉害吧！

谁都会有"不
小心忘了改"
的时候"

原来如此！！
这么说还真是
方便呐！

嗯

820G

我们称之为
"规范化"！！

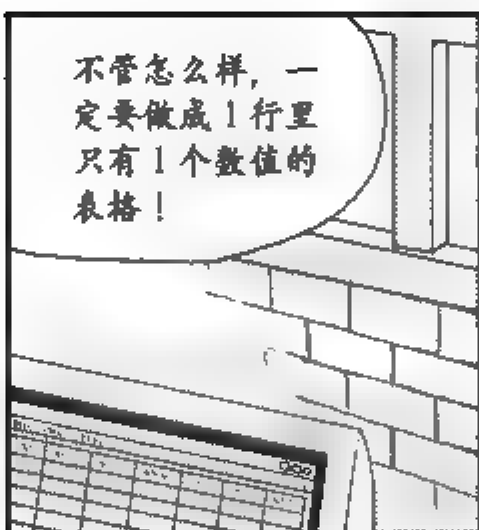
规范化
规范化
规范化

是的，
这里是重点。

像这样为了使数据不
发生矛盾将表格分开
的情况，

那么，
该怎么办
才好呢？

首先，



要分成"日期"、"出口国编码"、"出口国名称"的表格

和"商品编码"、"商品名称"、"单价"、"数量"的表格。

是不是啊？

销售表(第一范式①)

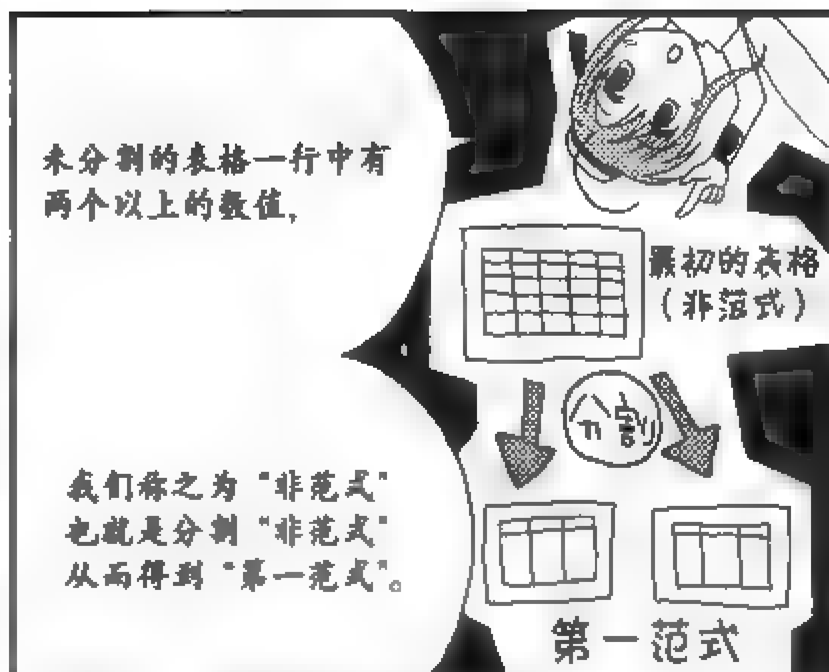
报表编码	日期	出口国编码	出口国名称
1101	3/5	12	米纳米王国
1102	3/7	23	阿尔法帝国
1103	3/8	25	理陀儿王国
1104	3/10	12	米纳米王国
1105	3/12	25	理陀儿王国

销售明细表(第一范式②)

报表编码	商品编码	商品名称	单价	数量
1101	101	香瓜	800G	1100
1101	102	草莓	150G	300
1102	103	苹果	120G	1700
1103	104	柠檬	200G	500
1104	101	香瓜	800G	2500
1105	103	苹果	120G	2000
1105	104	柠檬	200G	700

报表编码在两张表格中都有啊！

注意，这里是重点！！
这样分割开的表格叫做“第一范式”。
第一范式
第一范式
够了啦……


未分割的表格一行中有两个以上的数值，
我们称之为“非范式”也就是分割“非范式”从而得到“第一范式”。
最初的表格（非范式）
分割
第一范式

嗯？
等一下！

既然有第一，那一定也有第二、第三喽？
正确！

第一范式仍然不能作为关系数据库的表格来使用的 …
是这样啊……
抓紧了！！
第一范式
关系数据库之峰

来，让我们先看第一范式②！

嘭

是销售明细表。

报表编码	商品编码	商品名称	单价	数量
1101	101	香瓜	800G	1 100
1101	102	草莓	150G	300

"销售明细"表(第一范式②)

用这张表格仍然无法正确地管理商品。

啊？
为什么？

例如，一个都没卖出的商品"橙子"到货的话，

是不能往这个表格里添加的！

什么意思？

啊
是啊！

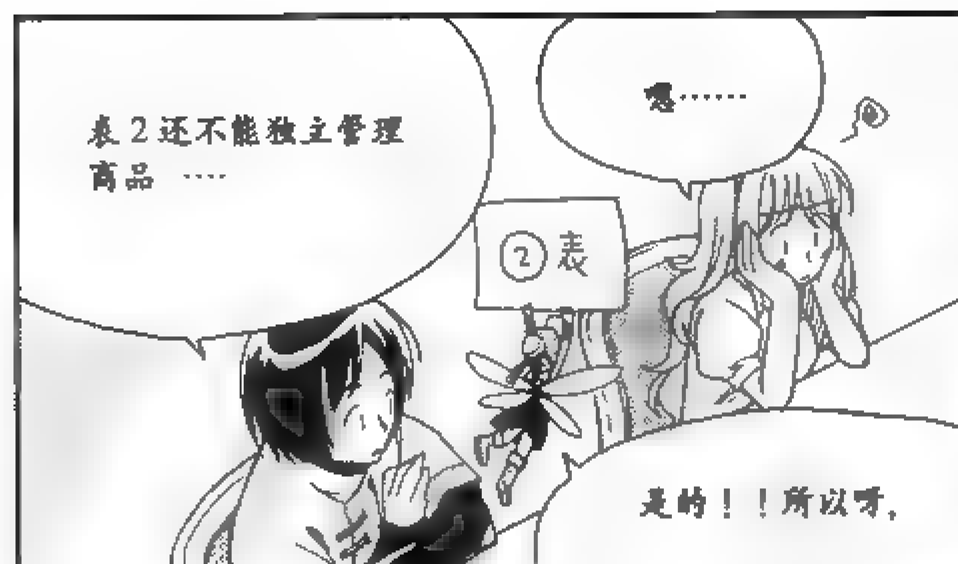
因为还没有销售，所以就没有销售报表编码和数量。

表②中关于商品的数据和销售相关的数据混在一起了。

非常好

啪

就是这样的！



这就是由之前的"第二范式②"分成的两个表格！

商品表(第二范式①)

商品编码	商品名称	单价
101	香瓜	800G
102	草莓	150G
103	苹果	120G
104	柠檬	200G

表1是关于商品的数据，

商品编码列的数值一旦确定，其他的如商品名称、单价等的值也就确定了。

销售明细表(第二范式②)

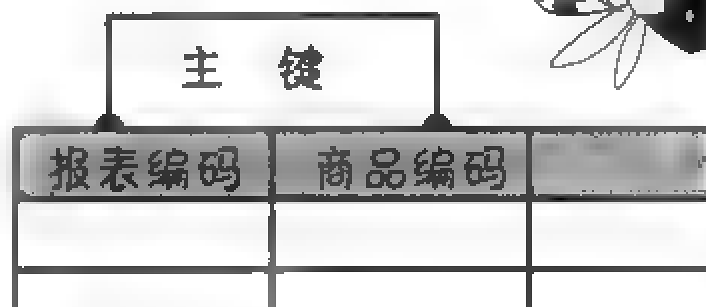
报表编码	商品编码	数量
1101	101	1100
1101	102	300
1102	103	1700
1103	104	500
1104	101	2500
1105	103	2000
1105	104	700

表②中，关于商品明细项目的数据，

也可以通过主键确定其他列的数值，

不过……

表②中，我们要将“报表编码”+“商品编码”这个组合看做主键。



两个商品有可能同时售出……

虽然是相同的商品，但售出的数量可能不一样……

也就是说，

我们要按“主键的值可以确定其他列的值”这一原则来分割表格。

表 1　商品编码　商品名称　单价

主键

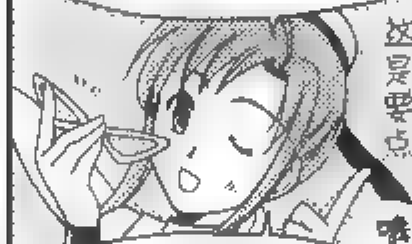
表 2　报表编码　商品编码　数量

主键

是这样的。

原来如此！

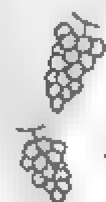
按照这个规则分割的表格，



我们称之为第二范式。

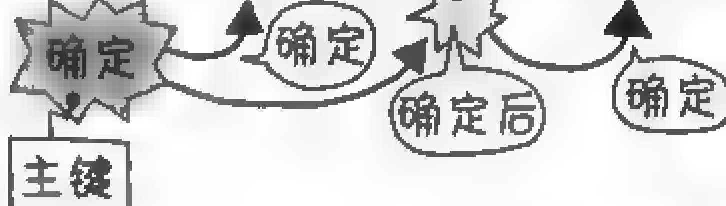
之前的橙子可以添加到第二范式①中了。

改变香瓜的价格只需要更改一行就可以了！



销售表（第一范式①）

报表编码	日期	出口国编码	出口国名称
1101	3/5	12	米纳米王国
1102	3/7	23	阿尔法帝国
1103	3/8	25	理陀儿王国
1104	3/10	12	米纳米王国
1105	3/12	25	理陀儿王国



这个表格的主键——报表编码一经确定，"日期"、"出口国编码"、"出口国名称"就全部确定了。

真的耶！！

销售表(第一范式①)

报表编码	日期	出口国编码	出口国名称
		12	米纳米王国
			阿尔法帝国
			理陀儿王国

是的，"第一范式①"可以就这么……

销售表(第二范式③)

报表编码	日期	出口国编码	出口国名称
		12	米纳米王国
			阿尔法帝国
			理陀儿王国

看做是"第二范式③"。

我们再来看看之前的
"第二范式③"？

销售表

嗯？

这个表格是不能管理出口国的。

哦？

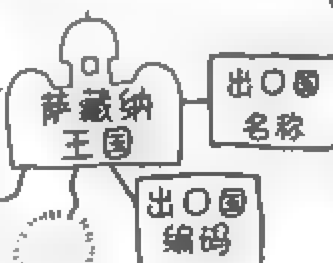
……

啊！

销售表（第二范式③）

报表编码	日期	出口国编码	出口国名称
1101	3/5	12	米纳米王国
1[illegible]	3/7	23	阿尔法帝国
[illegible]	3/8	25	理陀儿王国

一个水果都没有进口的"萨藏纳王国"是不能添加到这个表格中进行管理的。



表③中，
出口国相关的数据和销售相关的数据混在一起了……

嗯

想要单独管理出口国该怎么办呢？

对了！！

把它分割开来！

销售表（第三范式①）

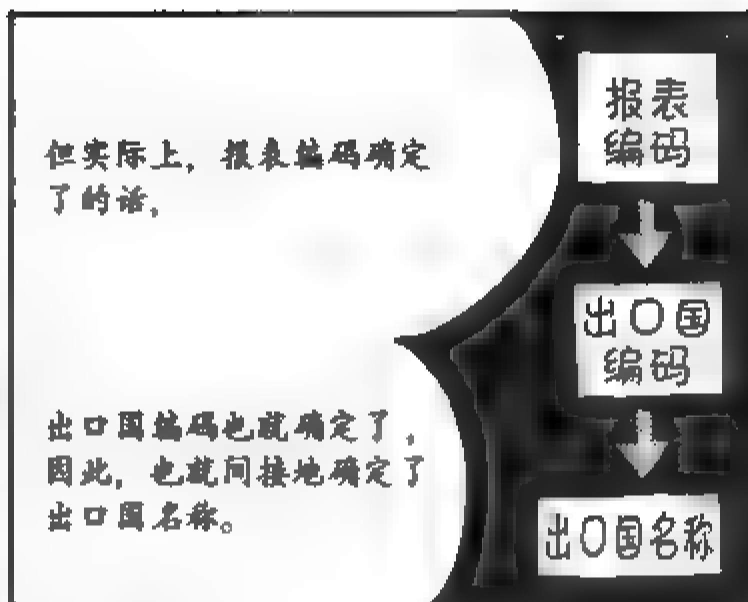
报表编码	日期	出口国编码
1101	3/5	12
1102	3/7	23
1103	3/8	25
1104	3/10	12
1105	3/12	25

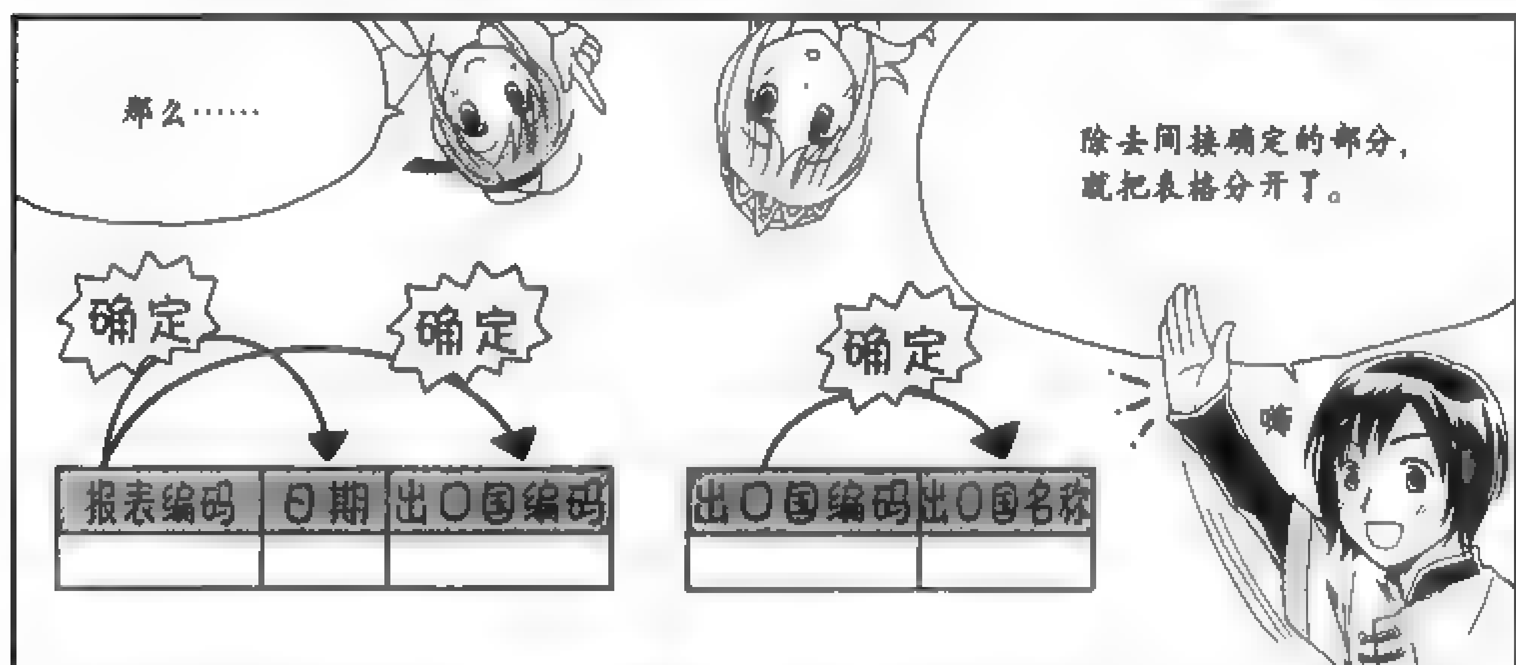
出口国表（第三范式②）

出口国编码	出口国名称
12	米纳米王国
23	阿尔法帝国
25	理陀儿王国

看——

之前的第二范式③中，出口国名称是通过报表编码确定的。
嗯！


但实际上，报表编码确定了的话，
出口国编码也就确定了，因此，也就间接地确定了出口国名称。
报表编码
出口国编码
出口国名称


那么……
除去间接确定的部分，就把表格分开了。
嗨
确定
确定
确定
报表编码
日期
出口国编码
出口国编码
出口国名称

是的！
这种由主键唯一确定其他列的表格叫做
第三范式！！

终于到第三了！
可以管理萨鲁纳王国了！！
啊哈——
第三正范式
成功了！！

销售表

报表编码	日期	出口国编码
1101	3/5	12
1102	3/7	23
1103	3/8	25
1104	3/10	12
1105	3/12	25

出口国表

出口国编码	出口国名称
12	米纳米王国
23	阿尔法帝国
25	理陀儿王国

销售明细表

报表编码	商品编码	数量
1101	101	1100
1101	102	300
1102	103	1700
1103	104	500
1104	101	2500
1105	103	2000
1105	104	700

商品表

商品编码	商品名称	单价
101	香瓜	800G
102	草莓	150G
103	苹果	120G
104	柠檬	200G

这样分割，可以用每个表格来管理商品、出口国和销售了。

商品

出口国

销售

也就能够顺利地管理商品和出口国了！

嗯，嗯！

即使添加数据也不会出现不一致的数据！！

终于可以松口气了……

最初的表格被一点一点地分割，

基础数据全部进入了其中某个表格里了呀！

销售明细表

出口国表

销售表

商品表

由销售报表制成的表格

表格显示了数据之间的关系。

确实是“关系”！！

销售表

报表编码	日期	出口国编码

参照

出口国表

出口国编码	出口国名称
12	米纳米王国
23	阿尔法帝国
[illegible]	理陀儿王国

就像销售表中的出口国编码参照出口国表中的出口国编码一样，

销售明细表中的商品编码，也需参照商品表中的商品编码。

销售明细表

参照

报表编码	商品编码	数量
1101	101	[illegible]
1101	102	[illegible]
1102	103	1900
1103	104	500
1104	101	2500
1105	103	[illegible]
1105	104	[illegible]

参照

商品表

商品编码	商品名称	单价
101	香瓜	800G
102	草莓	150G
[illegible]	[illegible]	120G
[illegible]	[illegible]	200G

销售明细表的报表编码参照销售表中的报表编码。

真想快点使用数据库
管理出口事务呀！

嗯~

好累啊~

是啊……

公主……
凯思殿下……

啊，爷爷！！！

在做什么呢
……

刚才就看见你们两个鬼鬼
祟祟的，
在干什么呢……

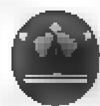
什么东西？

啊，那个，
当然是有原
因的……

咔嚓

凯思殿下，
没有教公
主一些奇
怪的东西
吧？

嗯，没有啊！



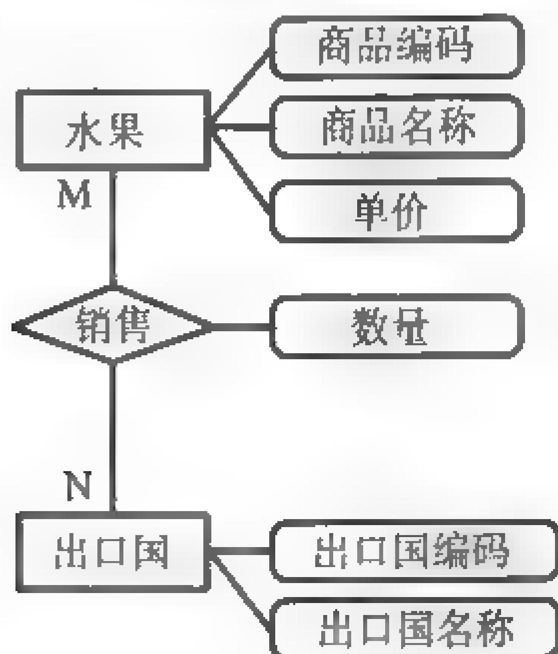
E-R模型

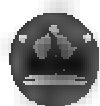
露娜公主和凯恩使用E-R模型掌握了王国的水果出口现状。首先，当然是要先掌握作为数据库管理对象的现实世界了。

使用E-R模型的分析，从现实世界抽取了**实体**(Entity)。实体就是像水果、出口国这些在现实世界存在的事物。从实体入手，自然就可以很容易地分析对象了。

另外，通过E-R模型的分析，还可以掌握实体之间的关系(Relationship)。通过分析实体之间的关系,就可以把握现实世界了。露娜公主和凯恩就是把握了水果和出口国之间的“销售关系”从而，进行分析的。

编码王国的水果出口到多个国家。另外，每个国家都会进口多种水果。因此，使用E-R模型分析出水果和出口国之间存在着多对多的关系，可以认为是M种水果对应N个出口国。这种实体之间的对应个数我们称做**基数**(Cardinality)。



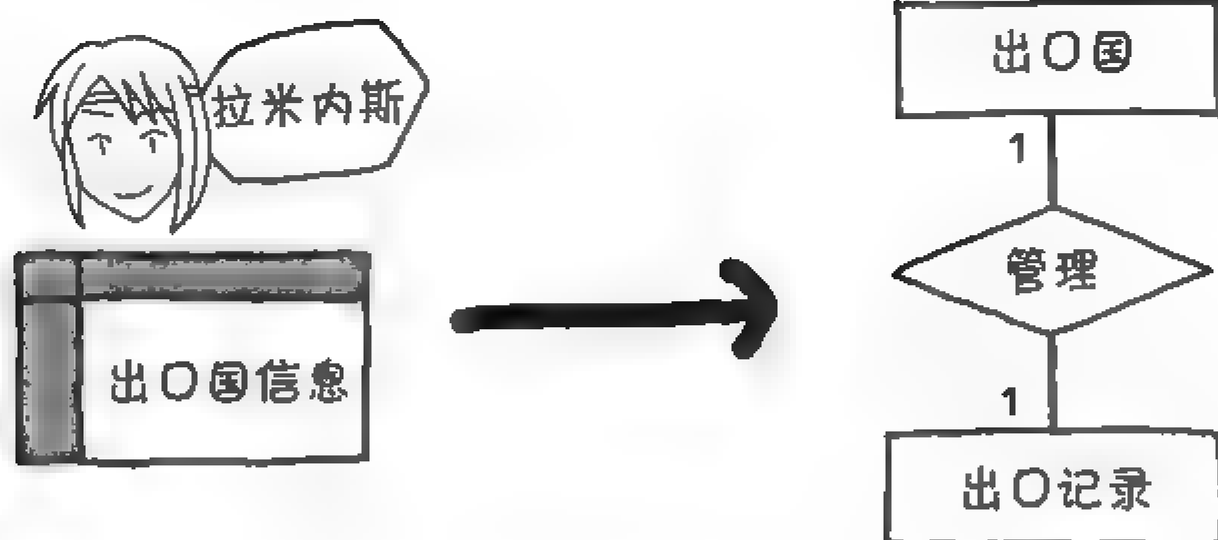


E-R模型的分析方法

那么，以下情况应该怎样进行分析呢？请思考一下。

事例1：一对一的关系

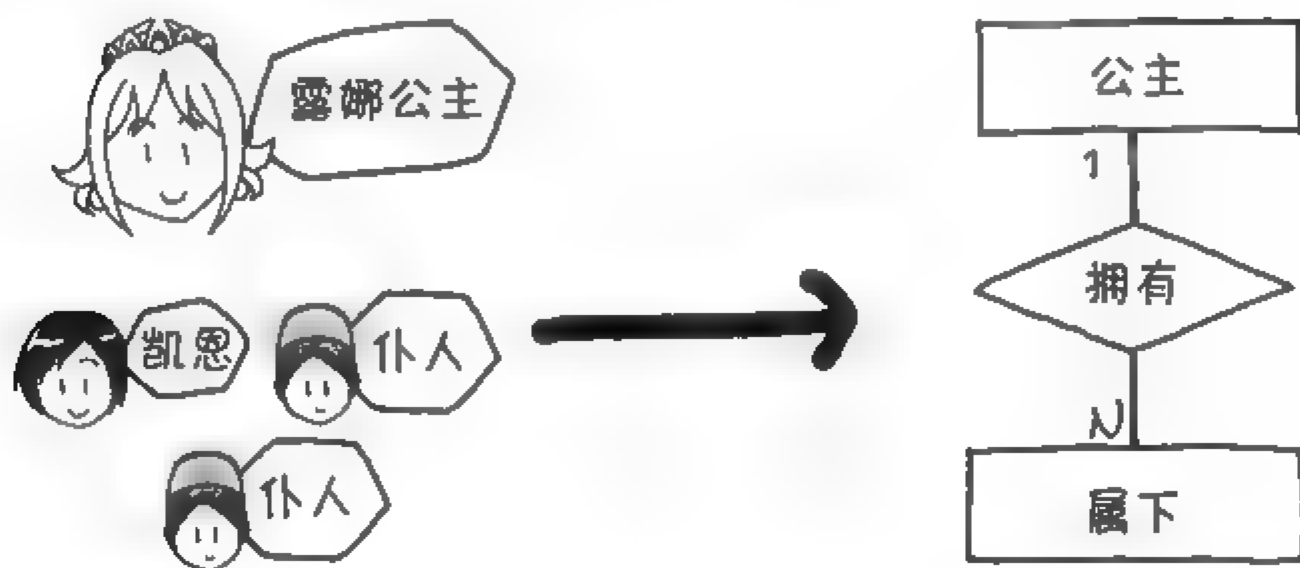
一个出口国管理一个出口记录信息。



这样的关系我们称之为一对一的关系 (one to one)。

事例2： 一对多的关系

露娜公主有多个仆人，这些仆人不属于其他的公主或国王。

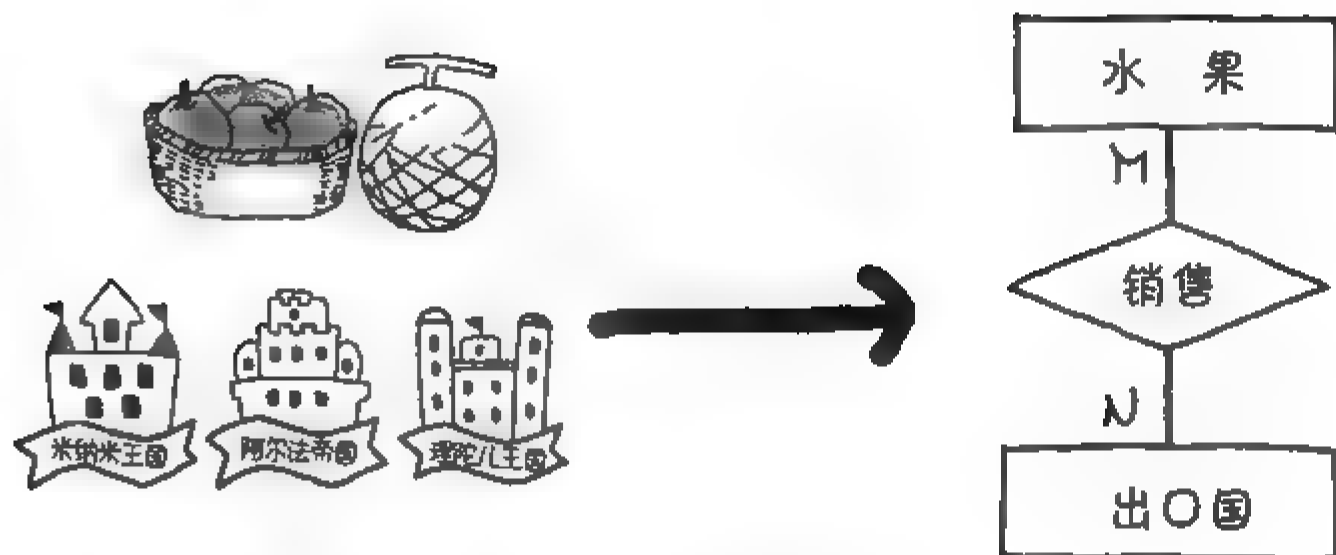


这样的关系我们称之为一对多的关系 (one to many)。

事例3： 多对多的关系

水果销往多个出口国。

出口国购买多种水果。

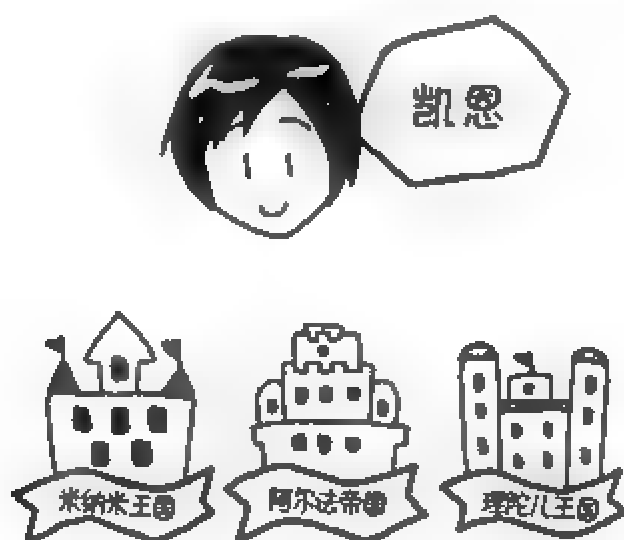


这样的关系我们称之为多对多的关系 (many to many)。

试着用E-R模型来分析

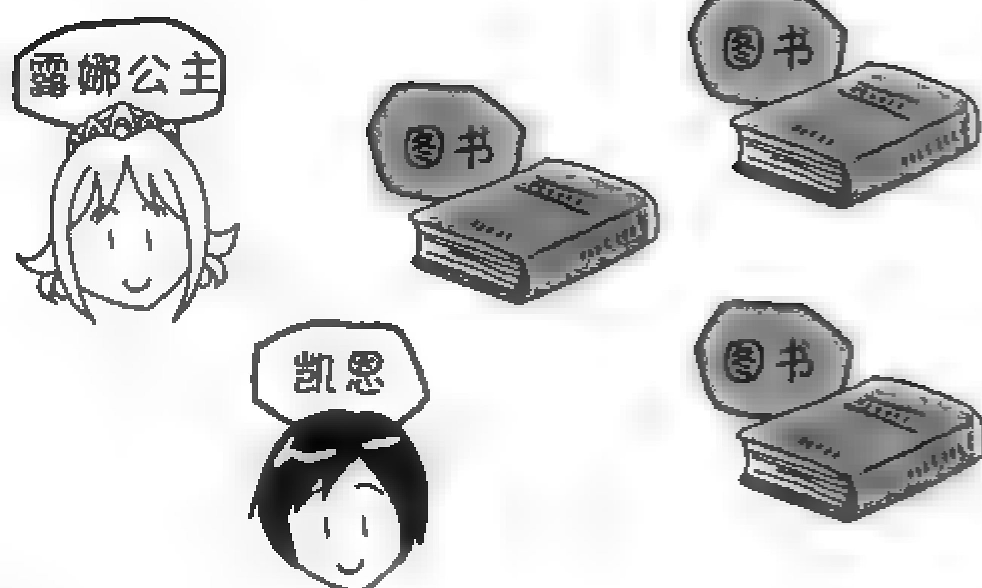
参照上面的事例分析，画出下面事例的 E-R 图。

一名职员负责多个客户。一个客户不会由两个或两个以上的职员来负责。



学生可以借出多本图书。

图书可以借给多个学生。

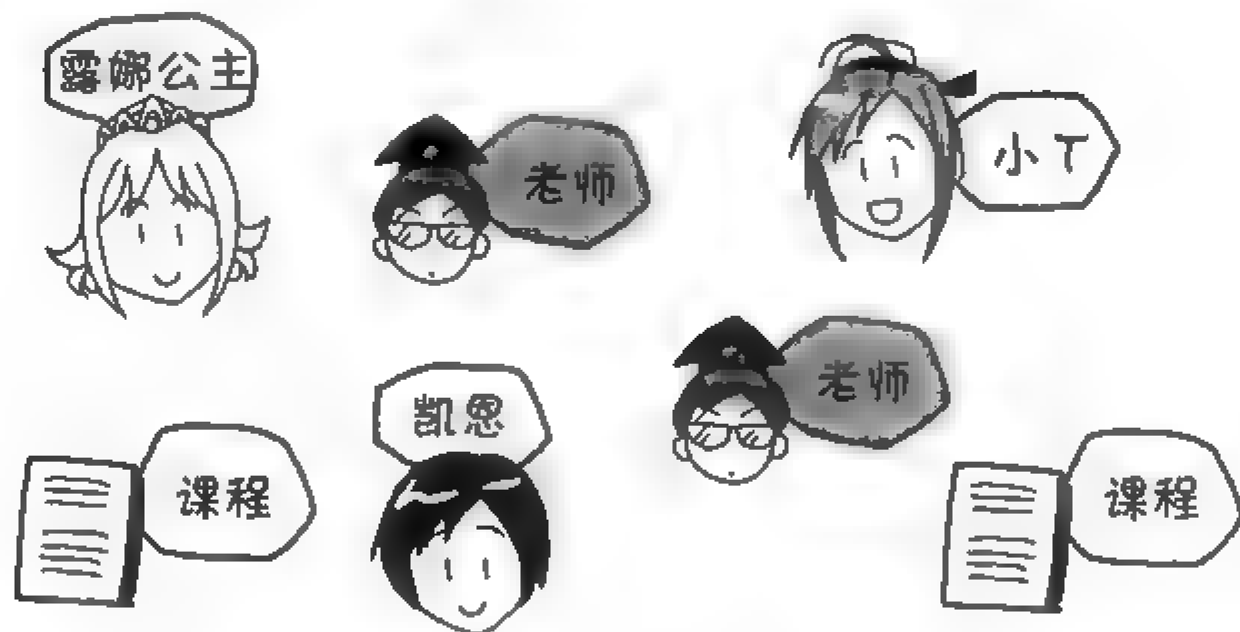


学生上多个科目的课程。

一个课程有多个学生。

一个老师教授多门课程。

一个课程由一个老师负责。

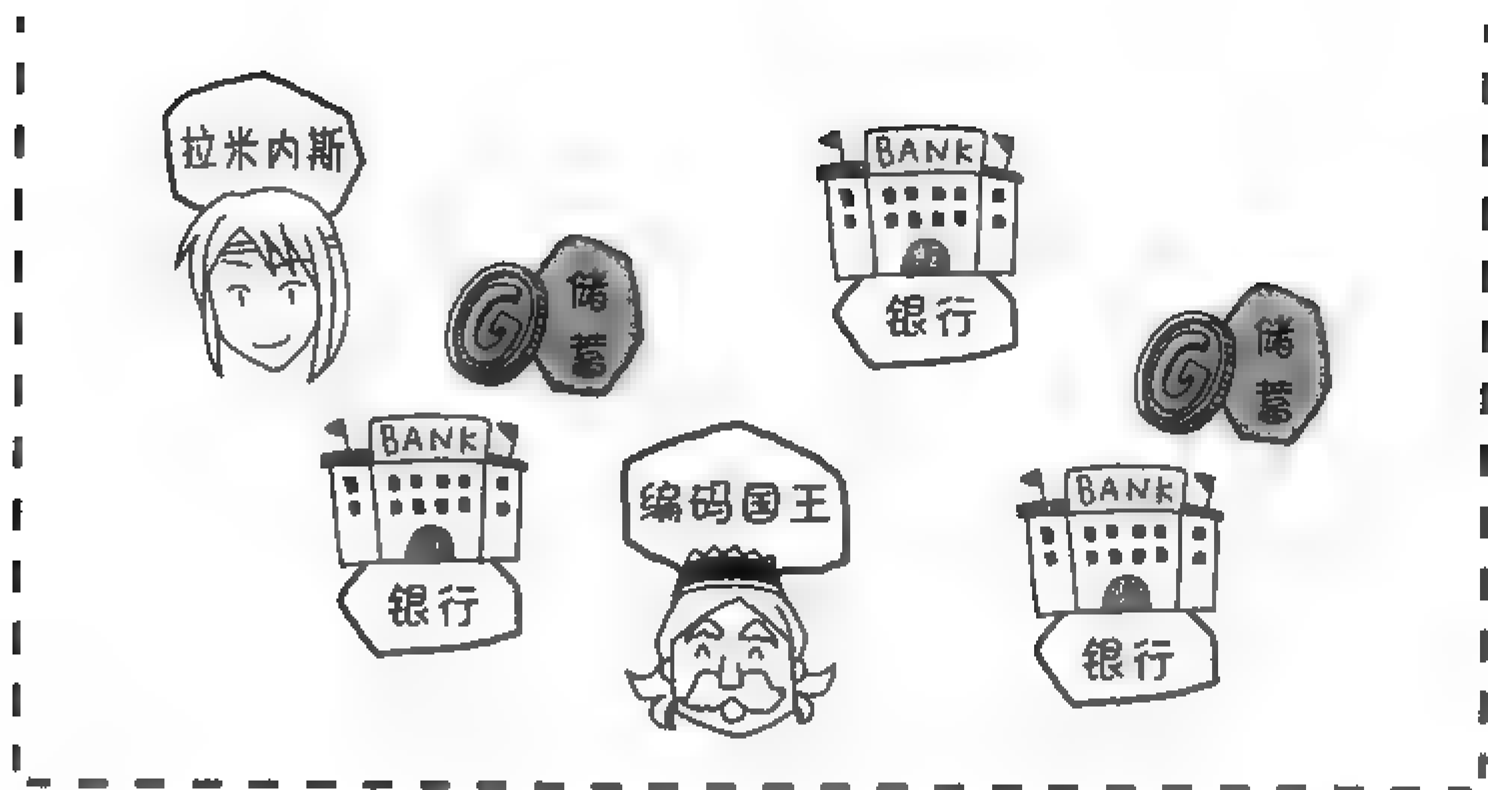


顾客可以开设多个储蓄账户。

一个账户只能由一个顾客开设。

各银行管理着多个储蓄账户。

一个账户只能由一个银行管理。



用E-R模型分析出来的结果不一定只有一个。因为观察世界的方法多种多样，我们只要按照自己的方法来做就可以了。

通过E-R模型的分析，能够把握现实世界。当然也就能够将沉默的现实世界套入模型进行分析了。

表格的规范化

露娜公主和凯恩后来学习了规范化(normalization)。规范化就是将现实世界落实在关系数据库表格里的工作。为了使用关系数据库正确管理从现实世界提取的数据，规范化的工作就非常必要。我们将本章中的规范化工作总结如下(阴影部分为主键)：

▼非范式

报表编码	日期	出口国编码	出口国名称	商品编码	商品名称	单价	数量

▼第一范式(first normal form)

报表编码	日期	出口国编码	出口国名称

报表编码	商品编码	商品名称	单价	数量

▼第二范式(second normal form)

报表编码	日期	出口国编码	出口国名称

报表编码	商品编码	数量

商品编码	商品名称	单价

▼第三范式(third normal form)

报表编码	日期	出口国编码

出口国编码	出口国名称

报表编码	商品编码	数量

商品编码	商品名称	单价

非范式是没有除去数据重复的表格。关系数据库中是不能使用这种表格来进行数据管理的。因此需要对其做分割表格的规范化工作。

第一范式 (first normal form) 将表格分割为单纯的二元表格，即一栏中只有一个项目，每一列都是不可分割的基本数据项。分割表格时除去了重复项目。

第二范式 (second normal form) 是按照通过可识别数据的键来确定其他列值的原则分割表格。这样，通过主键确定其他列的数值。

关系数据库中这种“通过某一列的值确定其他列的数值”的原则我们称之为**函数依赖** (functionally dependant)。第二范式是根据主键和其他列之间的函数依赖关系分割表格的。

第三范式 (third normal form) 是按照只能由主键确定其他列值的原则分割的表格。在关系数据库的函数依赖中，“通过某一列的值间接确定其他列的值”我们称之为**传递依赖** (transitively dependant)。第三范式是去除传递依赖而分割表格得到的。

试着规范化

以下情况该如何进行规范化呢？请思考一下。

下表如 Q2 所述，是管理借书的表格。他们被规范化到第几层了？

借出编号	日期	学生编号	学生名称	学生住址	院系	入学年度

书籍编号	书名	作　者	出版日期	总页数

借出编号	书籍编号	册数

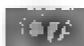
下表是管理借书的表格。他们被规范化到第几层了？

借出编号	日期	学生编号

学生编号	学生名称	学生住址	院系	入学年度

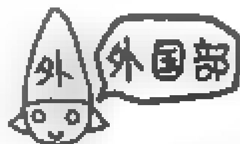
书籍编号	书名	作者	出版日期	总页数

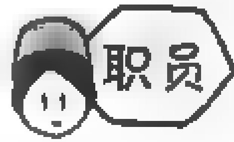
借出编号	书籍编号	册数



下表是记录各部门所属职员每月销售额的表格。一个部门有多个职员，职员不能属于多个部门。请规范化至第二范式。

职员编号	职员姓名	月	职员销售额	部门编号	部门名称





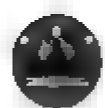
下表显示的是订单系统，请规范化至第三范式。注意，一个订单只能处理一个顾客。另外，一个订单编号能够处理多个商品。一个订单只能由一个人来负责。

订单号	日期	顾客编号	顾客名称	商品编号	商品名称	单价	负责人编号	负责人	数量

下表显示的是订单系统，请规范至第三范式。但是，商品是按照商品品类来划分的。

订单号	日期	顾客编号	顾客名称	商品编号	商品名称	单价	商品品类编号	商品品类名称	数量

针对各种各样的状况都能够设计出一套关系数据库表格，这一点，非常重要。



设计数据库

这里我们学完了设计数据库的知识。但是，数据库的设计并未就此结束。我们还需要学习数据库内部文件构造的详细设计知识及数据库的录入、显示方法等相关知识。

通常作为设计数据库的步骤，可以分为“概念模式”、“内部模式”和“外部模式”三部分。

概念模式（conceptual schema）是指将现实世界模型化的阶段进而，是确定数据库理论结构的阶段。概念模式的设计是通过E-R模型把握现实世界，进而规范化表格来实现的。

内部模式（internal schema）是从计算机内部看到的数据库，是确定数据库物理构造的阶段。内部模式的设计通过设计数据库高速检索方法来实现。

外部模式（external schema）是从用户和应用的角度来观察的数据库。外部模式的设计是通过设计应用程序所必要的数据来实现的。

内部模式 概念模式 外部模式

露娜公主和凯恩在本章以概念模式为中心设计了数据库，但仍需要继续学习以便更好地设计数据库。

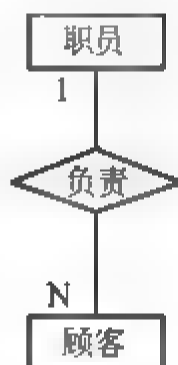
基础的数据库设计已经完成了，在下一章我们马上就要开始学习如何使用数据库了。

小结

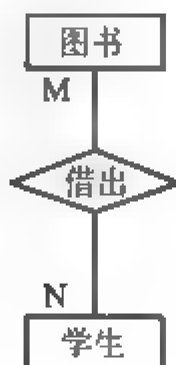
- E-R 模型用来分析实体和关系。
- 实体之间的关系数分为一对一、一对多和多对多。
- 关系数据库的表格有必要规范化。
- 关系数据库的设计分为概念模式、内部模式和外部模式三个部分。

答案

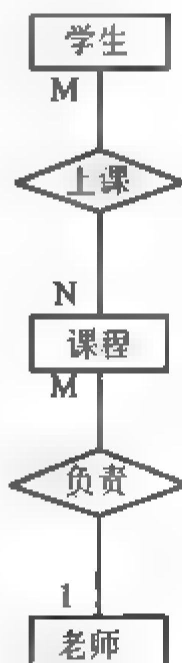
A1



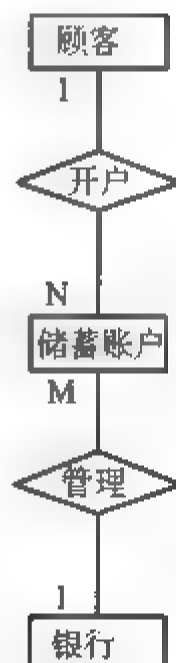
A2



A3



A4



A5 第二范式

A6 第三范式

A7

职员编号	月	职员销售额

职员编号	职员名称	部门编号

部门编号	部门名称

A8

订单编号	日期	负责人编号	顾客编号

顾客编号	顾客名称

订单编号	商品编号	数量

商品编号	商品名称	单价

负责人编号	负责人

A9

订单编号	日期	顾客名称

顾客编号	顾客名称

订单编号	商品编号	数量

商品编号	商品品类编号	商品名称	单价

商品品类编号	商品品类名称

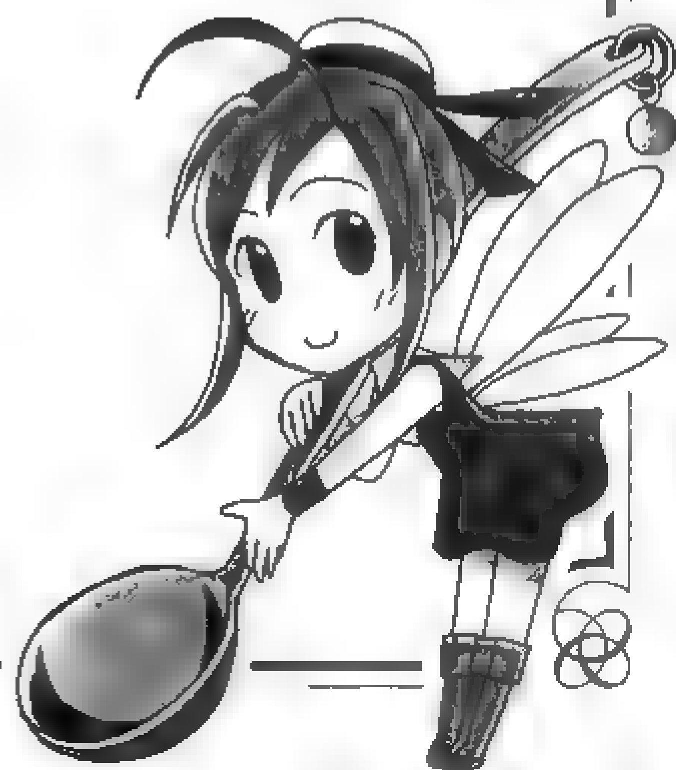
数据库的设计

在这一章，我们学习了数据库表格的设计方法。但是，数据库的设计方法不只限于一种。根据不同的分析、设计方法，数据库的便捷性和效率也会不同。因此，在设计阶段要好好考虑，设计合适数据库非常重要。

在设计数据的过程中，除了设计表格以外，还有很多工作要做。例如，要考虑表格中使用的数据类型。还要确定表示数值和货币的列及表示字符串的列。另外，还要设计能够进行高速检索的检索方法。有时还要一边考虑实体性的文件，一边进行设计。为了确保安全，还要控制可访问数据库的用户。在设计数据库时需要考虑的事情很多很多，这部分工作将在以后的章节中陆续介绍。

第4章

使用数据库——SQL的基本操作



这么走在大街上总能让我想起小时候……
啊哈哈！！
公主经常逃课到城里来。
公主！！
露娜公主！！
哒哒哒
逃课？
到货
应季桃子
擅自出城可不好啊！
呼
呼
呼

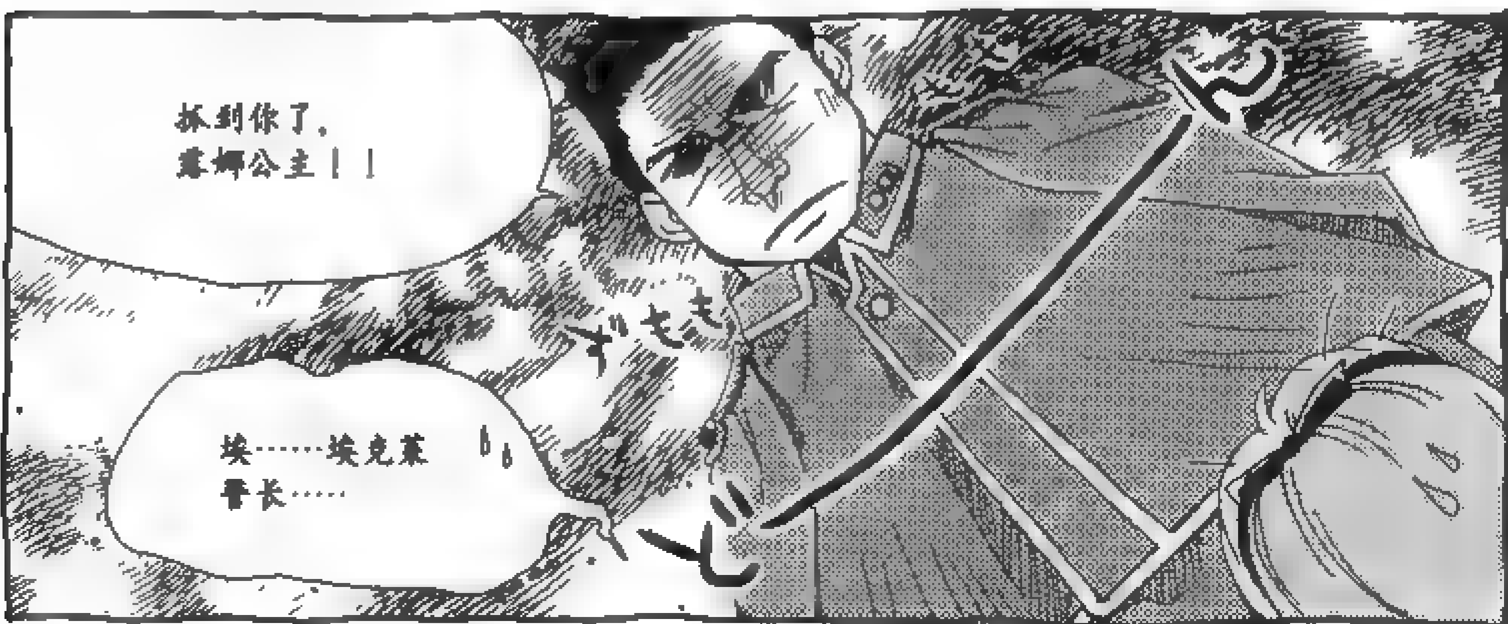
干吗嘛，凯恩！
如果教训我的话就
不要跟来好了！
哼
哼
可，
可是

作为一名随从，
跟随公主就是我
的责任。
责任感！

因为工作才
跟来的吗？
不是因为我们
的友情吗？
啊？
那个……

不理你了啦！
啊，不是那个
意思……

咚
哎
呀


抓到你了，
露娜公主！！
埃……埃克莱
警长……

顽皮过头了哟，
公主！！
呵……

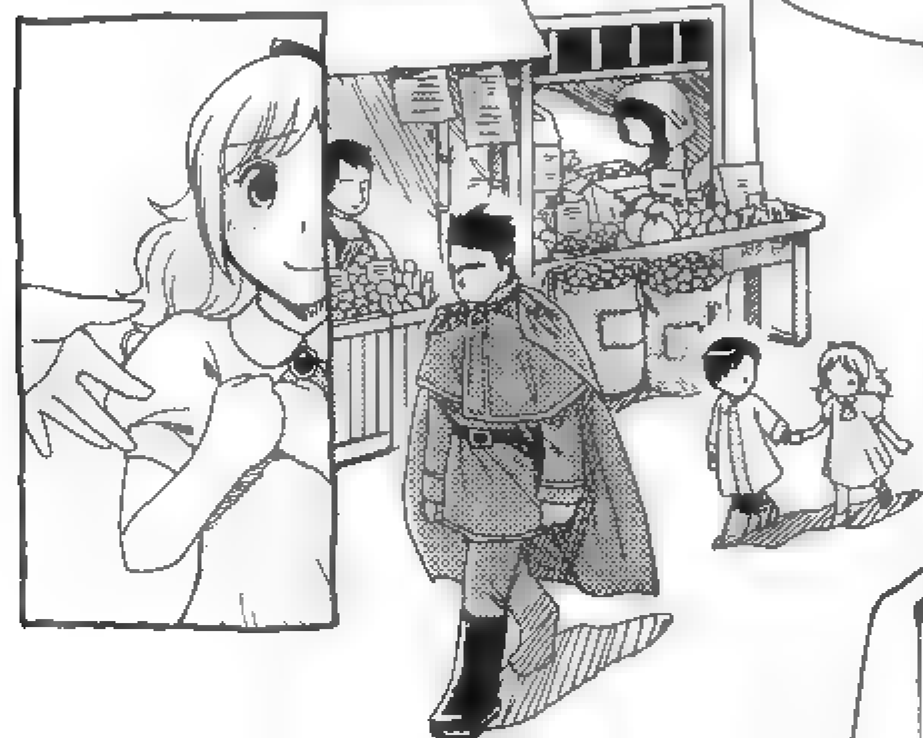
不是这样的！！
哼

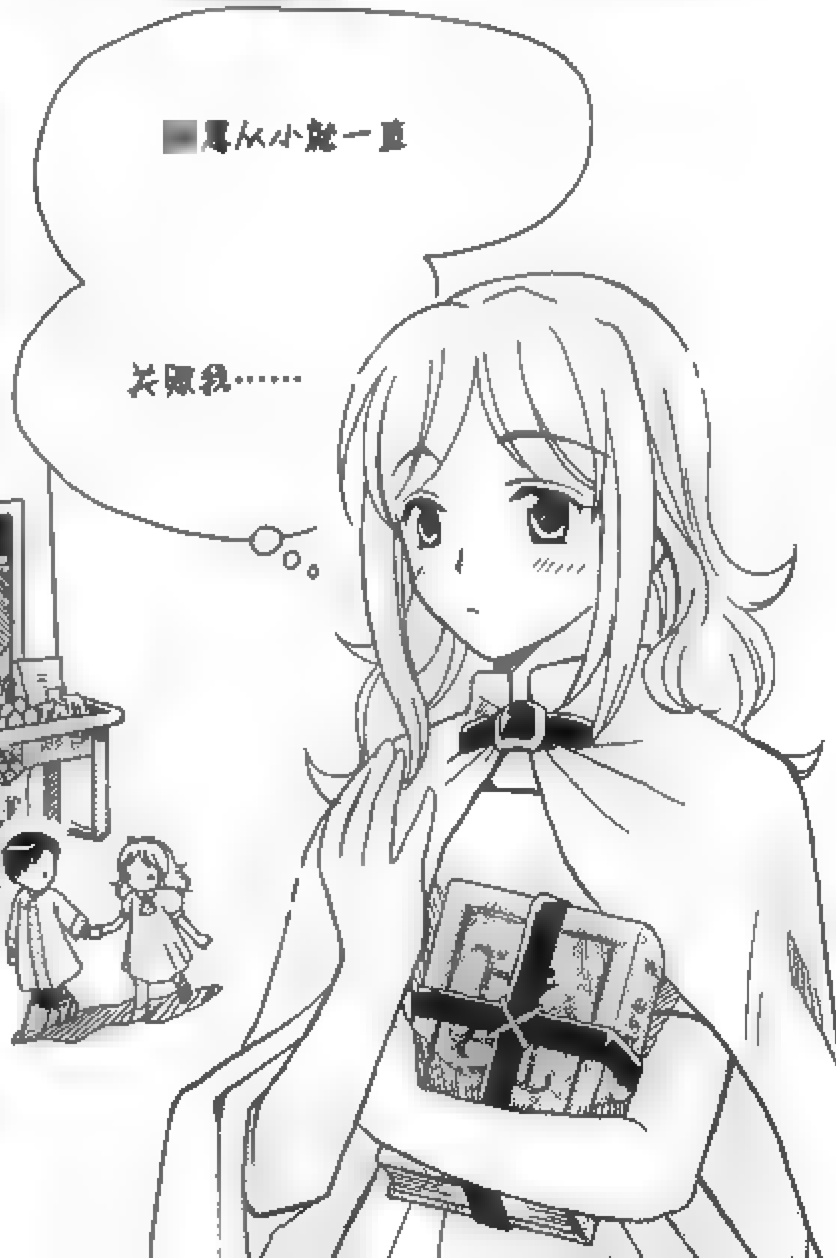
是……是我带露娜
公主出来的…
凯恩……

……

哼哼~

公主，
我们回去吧。

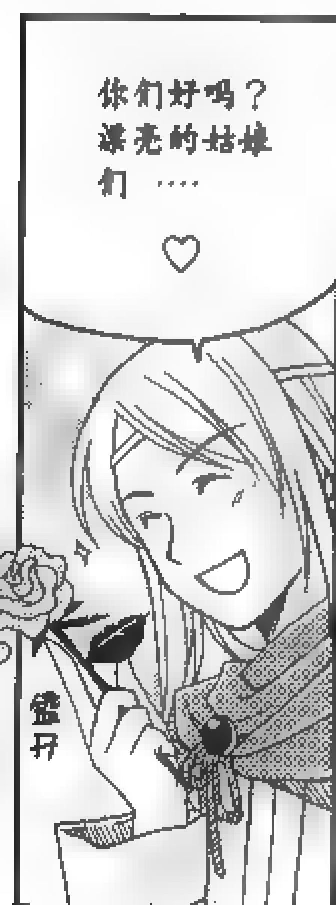



恩从小就一直
关照我……

怎么了？
啊！！
没什么。
？
什么都没有！！

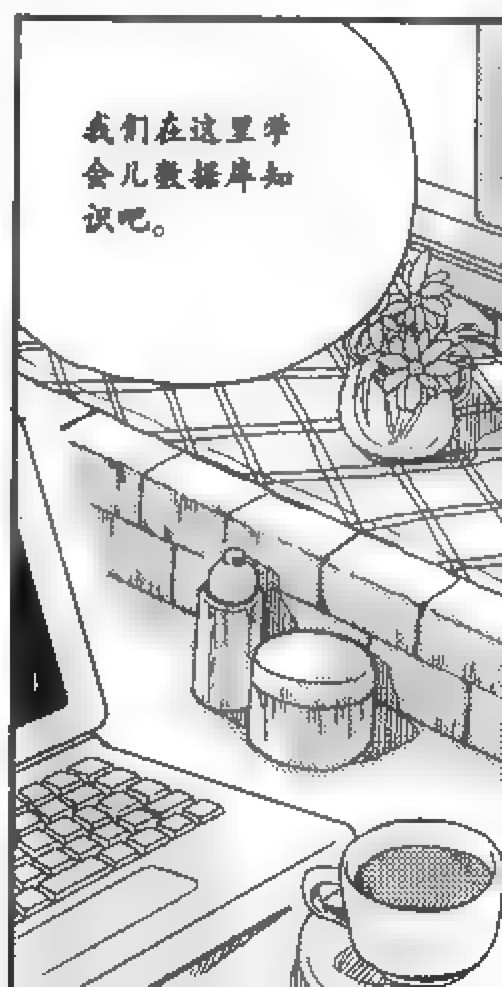
唉，真是的……
嗯？
！！
哈哈
啊！

拉米内斯！！
哈哈
拉米内斯



你们好吗？漂亮的姑娘们……
♡

坏了，他过来了。
凯恩！！我们进这家店吧！！
嗯？
哈哈哈
大家好
大家好



我们在这里学会儿数据库知识吧。

你们好
久等了!!
啊，外面真好啊！

哇！
怎么回事和平时不太一样？

我们学完基本的数据库设计了。
是啊！

那么，
接下来我们学习怎么实际应用所做的数据库吧！
终于……
要用了
这个表情真好

首先，
使用数据库必须将数据录入并调取必要的数据吧。
因此，
我们要学习 SQL。
SQL

SQL？
好像又是很难呀
是的

销售表

报表编码	日期	出口国编码
1101	3/5	12
1102	3/7	23
1103	3/8	25
1104	3/10	12
1105	3/12	25

出口国表

出口国编码	出口国名称
12	米纳米王国
23	阿尔法帝国
25	理陀儿王国

商品表

商品编码	商品名称	单价
101	香瓜	800G
102	草莓	150G
103	苹果	120G
104	柠檬	200G

销售明细表

报表编码	商品编码	数量
1101	101	1100
1101	102	300
1102	103	1700
1103	104	500
1104	101	2500
1105	103	2000
1105	104	700

这段时间我们设计了这些表格，对吧！

其实，制作这些表格和向数据库中录入数据都需要SQL。

■SQL的功能

- 生成表格
- 录入、调取数据
- 管理用户

使用SQL的话就可以与数据库进行对话，从而完成这些工作了。

这样的工作，那样的工作都可以～

能做很多事情啊！！

好厉害！

可是，会很麻烦吗？

还好了啦！！

不管怎样，总算是进步了，不是吗？

是啊是啊！

是吗……好吧，真想赶快用上数据库啊！

就是这种气势——

之前设计的表格和数据已经录入了。

啪

那么，试着调取数据吧！！

使用SELECT命令检索

使用 SQL 从这张表格中仅调取"商品名称"制成"商品名称列表"吧！

该怎么做呢？

"我想从商品表中调取商品名称列！"

拜托了

像这样，对着数据库许愿就可以了。

数据库先生

拜托了

原来如此。用 SQL 表示的话也行吧！

双手合十是不行的哟~

我知道了哦

是的！要用 SQL 语言。

拜托了

就是这样的！

SELECT 商品名称

FROM 商品；

SQL 的一个对话我们叫做命令！！

这个 SQL 命令是由"SELEOT 商品名称"和"FROM 商品"两个单词组成的。

这个单词组我们称之为短语。

在 SELECT 短语中指定想要调取的列名，在 FROM 短语中指定想要调取的表格名称。

商品表

商品编码	商品名称	单价
101	香瓜	800G
102	草莓	150G
103	苹果	120G
104	柠檬	200G

FROM

SELECT

调取的结果就是这个!!

这样就从商品表中把所有的商品名称全部调出来了！

商品名称
香瓜
草莓
苹果
柠檬

这就是用 SQL 进行对话，是吗？

是的！
通过应用各种各样的命令就可以调取需要的数据了。

应用啊……

嗯……

啊！！
那么举个例子。

想要生成"单价为 200G 以上的商品列表"，该怎么办呢？

这种情况下调取全部的商品数据是不行的，

要调取的只是单价在200G以上的商品。

是的！

也就是说，

这时要在“WHERE”短语中设定条件。

WHERE

WHERE 单价 >= 200

这样设定！！

原来如此。
可是……

总是一个一个地设定列名很不方便……

那也没问题！
想要设定全部列的时候……

呼

使用“*”就可以了！
写出整个句子

看

```
SELECT *
FROM 商品
WHERE 单价 >= 200
```

就是这样的！

“单价在200G以上的商品”就全都……

看

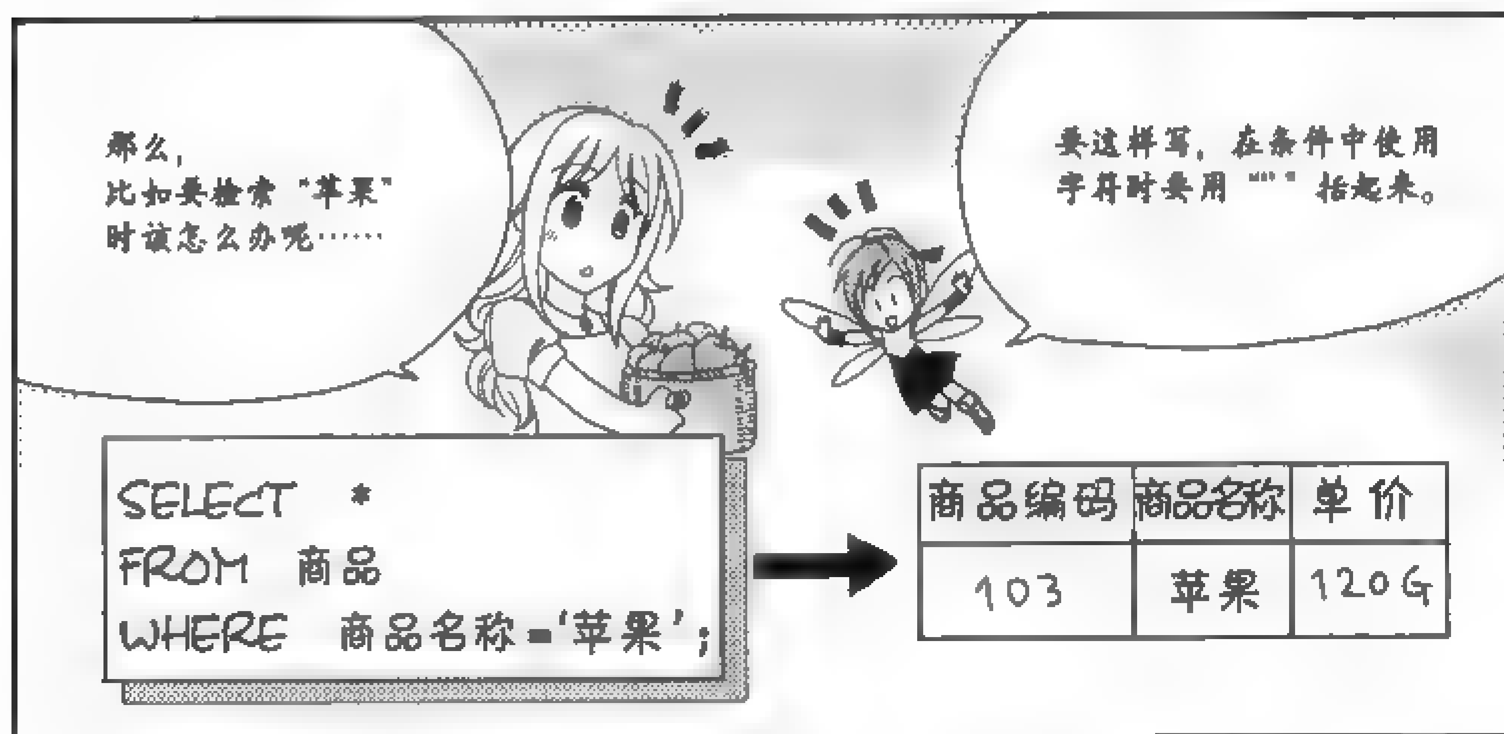
200G以上的商品

商品编码	商品名称	单价
101	香瓜	800G
104	柠檬	200G

调出来了

!!

换个条件就可以调取"单价不满200G的商品"
WHERE 单价 < 200
这么写!!
是的!

似乎需要研究一下设定条件的方法吗。
好像是耶!

那么,
比如要检索"苹果"时该怎么办呢……
要这样写,在条件中使用字符时要用"''"括起来。
SELECT *
FROM 商品
WHERE 商品名称='苹果';
商品编码 商品名称 单价
103 苹果 120G

这样就能够很便捷地检索"苹果"的数据了。
是这样的!

可是有时候我们可能不清楚商品名称哟

这个时候该怎么办呢？

这时要用LIKER和符号的组合。

用"%"表示任意多个的字符，就像这样……

检索商品名称最后一个字是"果"和"莓"的商品。

草莓 莓

苹果 果

```
SELECT *
FROM 商品
WHERE 商品名称 LIKE '%果'
OR 商品名称 LIKE '%莓';
```

商品编码	商品名称	单价
102	草莓	150G
103	苹果	120G

这样草莓和苹果就被检索出来了。

真方便啊！

对吧？

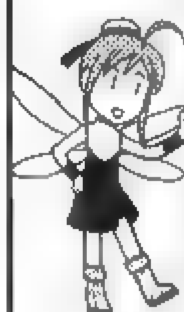
使用计算函数来计算

是啊是啊！
如果再加上 ORDER BY 短语还可以将检索结果排序呢！

把单价由低到高排序时加上“ORDER BY 单价”。

这样对商品就可以进行各种各样的查询了！

好厉害啊



```
SELECT *
FROM 商品
WHERE 商品名 LIKE '%果'
OR 商品名称 LIKE '%莓'
ORDER BY 单价;
```

商品编码	商品名称	单价
103	苹果	120G
102	草莓	150G

哇，
我还想知道更多关于 SQL 的知识，小 T！

越来越有趣了!!

啪

激动

呵呵，是吧？

好高兴啊~ 呵呵~~

那么……这个？

在“SELECT”短语中插入求各行平均值的“AVG（列名）”!!

```
SELECT AVG(单价)
FROM 商品;
```

看

这样就能算出来了，真让人吃惊！

看

AVG(单价)
317.5

就能知道商品的平均单价了哟！

平均价格原来是这么多啊……
以前还真不知道啊！
原来不知道……
虽然是自己的国家但不知道的事还很多呐……
就是这样的
SQL中还有计算调取数据的功能呢！
方便吧~~~
那么，
除了求平均数还有很多其他的计算吗？
当然了！！
行数、合计、平均、最大值、最小值这些和“AVG”一样，都可以通过设定“计算函数”来求得。

计算函数

计算函数	含义
COUNT(*)	求行数
COUNT(列名)	求取非空值行数
COUNT(DISTINCT 列名)	求取排除空值及重复行的行数
SUM(列名)	求取各行的合计值
AVG(列名)	求取各行的平均值
MAX(列名)	求取各行的最大值
MIN(列名)	求取各行的最小值

就是这样子的！
哇……

那么，
计算数商品的种类时？
使用“COUNT”！
推
COUNT
想知道价格最高的商品呢？
使用“MAX”！
这个！
MAX

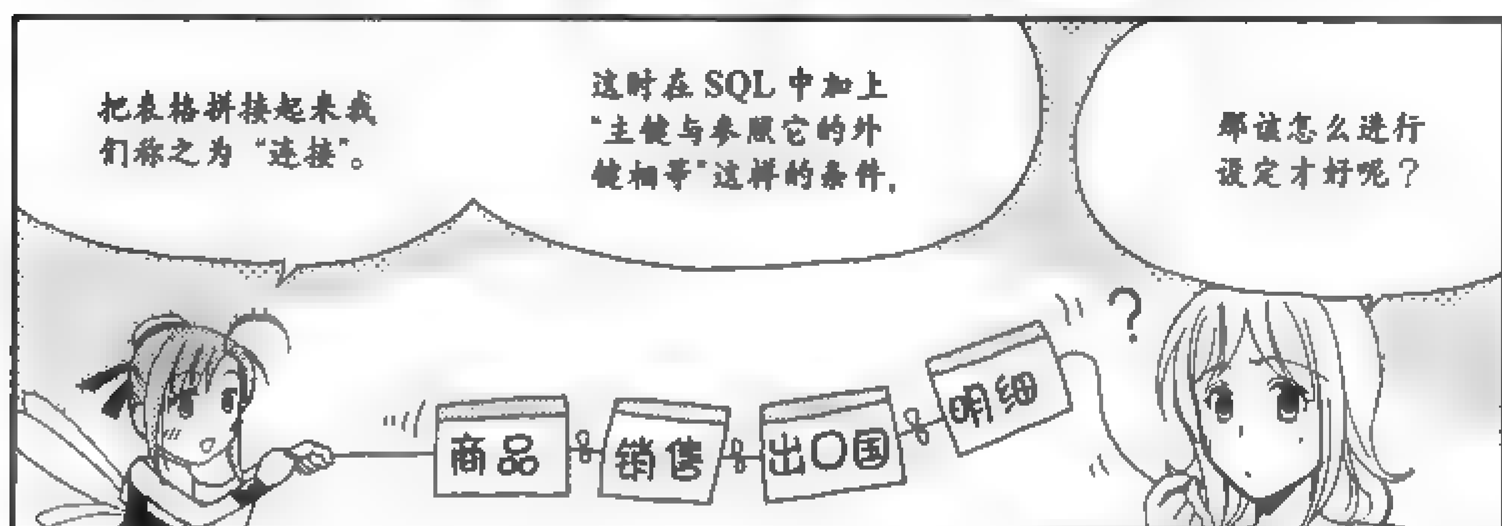
用 SQL 计算函数就可以知道很多事情了！
还能够将数据分组化计算哟！

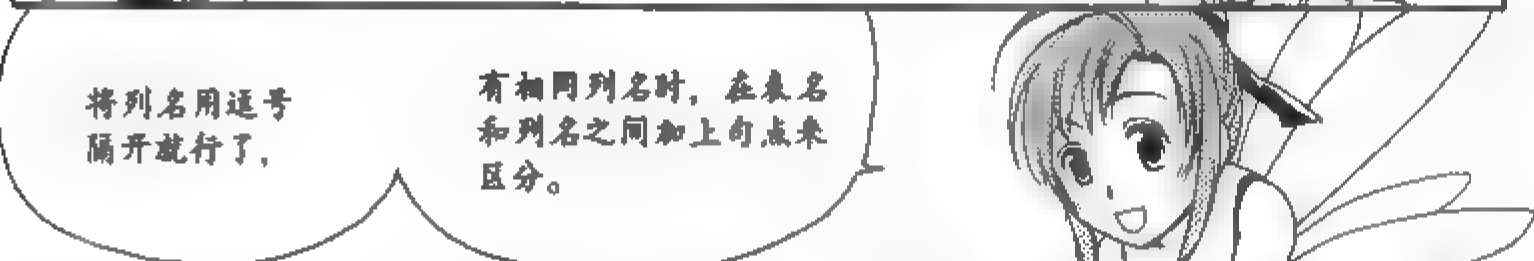
知道计算功能很厉害了！
可是……

我们现在使用的销售报表怎么办呢？
啊，又是那个奇怪的家伙……
那么，

这次我们试着用 SQL 来做现在使用的“销售报表”吧!!
咚！

连接表格






```
SELECT 销售.报表编码，日期，
       销售.出口国编码，出口国名称，
       销售明细.商品编码，商品名称，单价，数量
FROM   销售，销售明细，商品，出口国
WHERE  销售.报表编码＝销售明细.报表编码
        AND
       销售明细.商品编码＝商品.商品编码
        AND
       出口国.出口国编码＝销售.出口国编码；
```

设定成“表格名称.列名”的形式。

报表编码	日期	出口国编码	出口国名称	商品编码	商品名称	单价	数量
1101	3/5	12	米纳米王国	101	香瓜	800G	1100
1101	3/5	12	米纳米王国	102	草莓	150G	300
1102	3/7	23	阿尔法帝国	103	苹果	120G	1700
1103	3/8	25	理陀儿王国	104	柠檬	200G	500
1104	3/10	12	米纳米王国	101	香瓜	800G	2500
1105	3/12	25	理陀儿王国	103	苹果	120G	2000
1105	3/12	25	理陀儿王国	104	柠檬	200G	700

这样，尽管是独立管理商品、出口国和销售，但也可以随时调出销售报表的相关数据了！

和现在使用的一样耶！！

好厉害啊！！

哇哇——

生成表格

这么说这张表格也是小T用SQL做成的了？

表格和数据已经录入进去了，不是吗?

这个这个

是啊！

那是怎么做的呢？

CREATE TABLE

在制作表格时用"CREATE TABLE"命令。

```
CREATE TABLE 商品
(
  商品编码 NUMBER(3, 0),
  商品名称 CHAR(20),
  单价 NUMBER(10,0),
  PRIMARY KEY(商品编码)
);
```

商品编码	商品名称	单价

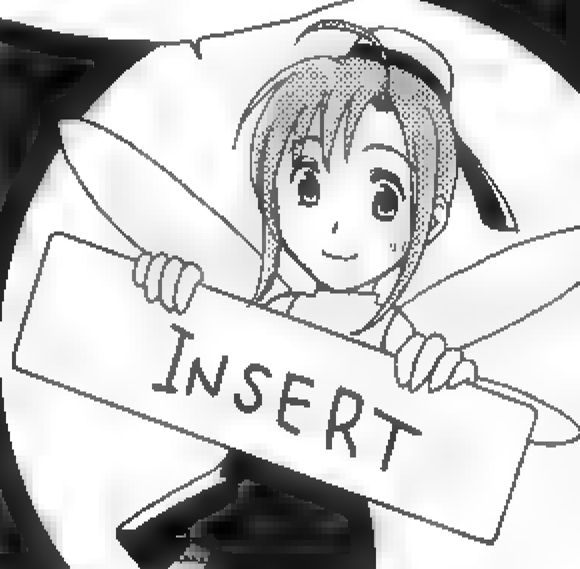
还要设定主键。这里将商品编码设定为主键，

就是这样的

也可以设定数据的值域。

可以防止输入错误的数值。

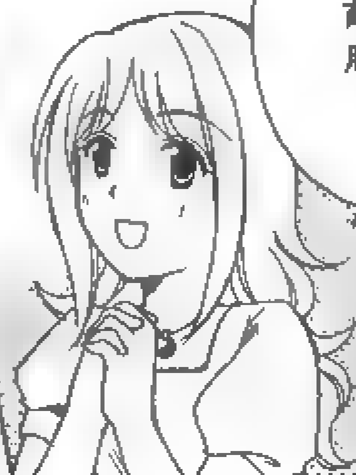
商品编码	商品名称	单价

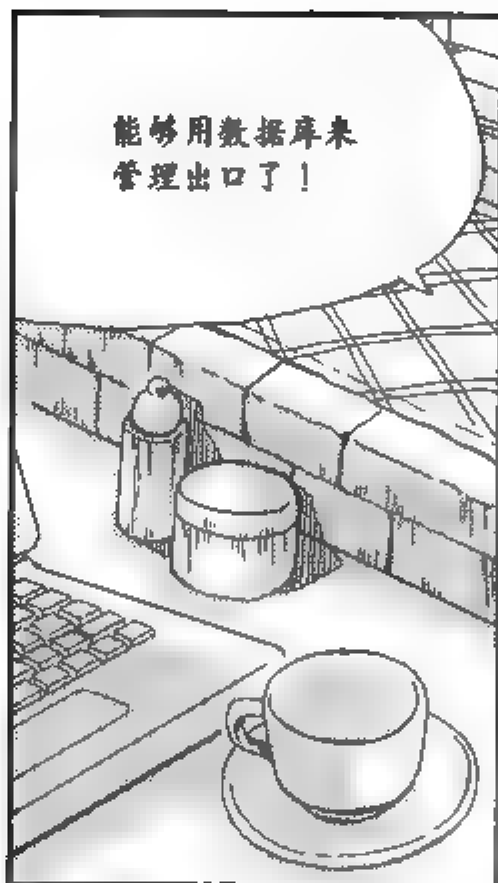



```
INSERT INTO 商品(商品编码, 商品名称, 单价)
VALUES(101, '香瓜', 800);
```

↓

商品编码	商品名称	单价
101	香瓜	800G



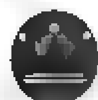

能够用数据库来
管理出口了！

SQL 的基础就是
SELECT 命令！
一定要牢记！
前面还有很
长的路要走
是啊
能够熟练使用 SQL 的
话，就可以随心所欲
地管理数据库了……
啊！
都这么晚了！

日落前我们
回城吧！
又要被埃克莱
尔骂了！！
啊……
对啊！

这么说以前经
常被骂呀……
啊
凯恩！！

快点回去了！
好的！



SQL的功能

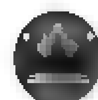
露娜公主和凯恩学习了 SQL 的诸多功能。SQL(Structured Query Language) 是用来操作关系数据库的数据库语言。SQL 就是具有以下功能的语言。

① 数据定义语言 DDL(Data Definition Language)……生成表格。
② 数据操作语言 DML(Data Manipulation Language)……输入、调取数据。
③ 数据控制语言 DCL(Data Control Language)……管理用户的访问等。

① 首先，SQL 有制成数据库结构的功能。例如，在数据库中生成事先设计好的表格，还可以变更和删除表格。具有这种功能的数据库语言我们称之为数据定义语言 (DDL：Data Definition Language)。

② 其次，SQL 具有操作数据库中的数据的功能。它可以检索数据的，另外，还具有向表格中插入、删除、变更数据的功能。具有这种功能的数据库语言我们称之为数据操作语言 (DML：Data Manipulation Language)。

③ 最后，SQL 具有控制数据库的功能。即便是多人同时使用数据也不会出现矛盾。具有这种功能的数据库语言我们称之为数据控制语言 (DCL：Data Control Language)。



使用SELECT命令检索

露娜公主和凯恩是从 SQL 中的数据检索功能开始学起的。SQL 是通过输入一组命令来检索数据的。

SQL 命令是由短语 (Phrase) 组合构成的。使用 SELECT 短语是最基本的 SQL 命令。例如，检索单价为 200G 的商品时使用如下 SQL 命令：

```
SELECT *
FROM 商品
WHERE 单价 = 200;
```

将短语组合构成 SQL命令

SELECT 命令中指定了“哪一列 (SELECT)”、“从哪个表格 (FROM)”、“以怎样的条件 (WHERE)”。组合短语对 SQL 进行查询 (query)。由于使用了比较直观的易于理解的询问形式的命令，所以即便是对处理数据库不很熟悉的人，也可以对一些必要的数据进行检索和使用。

使用比较运算符设定条件

另外，凯恩所说的“研究条件的设定方法”，是指在通过 SQL 进行查询时，要用心考虑条件 (condition) 的设定方法，才能够调取出符合要求的数据。那么在这里我们来更加详细地介绍一下条件的设定方法吧。

设定条件时使用的“> =”和“=”我们称之为“比较运算符”。即“A 大于等于 B”这样的条件使用“> =”来表示。另外，“A 等于 B”这样的条件用“=”来表示。

此外还有其他比较运算符，如下表所示。

■ 比较运算符

比较运算符	含义	示例	示例的含义
A = B	A 等于 B	单价 = 200	单价为 200G
A > B	A 大于 B	单价 > 200	单价高于 200G
A > = B	A 大于等于 B	单价 > = 200	单价在 200G 以上
A < B	A 小于 B	单价 < 200	单价不足 200G
A < = B	A 小于等于 B	单价 < = 200	单价在 200G 以下
A < > B	A 不等于 B	单价 < > 200	单价不为 200G

使用逻辑运算符制作条件

仅使用比较运算符制作的条件有时还不够用。这时就要用到一种叫做逻辑运算符 (logical operator) 的符号来设定条件了。使用逻辑运算符后，能够组合条件生成更加复杂的条件。逻辑运算符可以将用比较运算符设定的条件 A 和 B 进行组合，生成如下表所示的更复杂的条件。

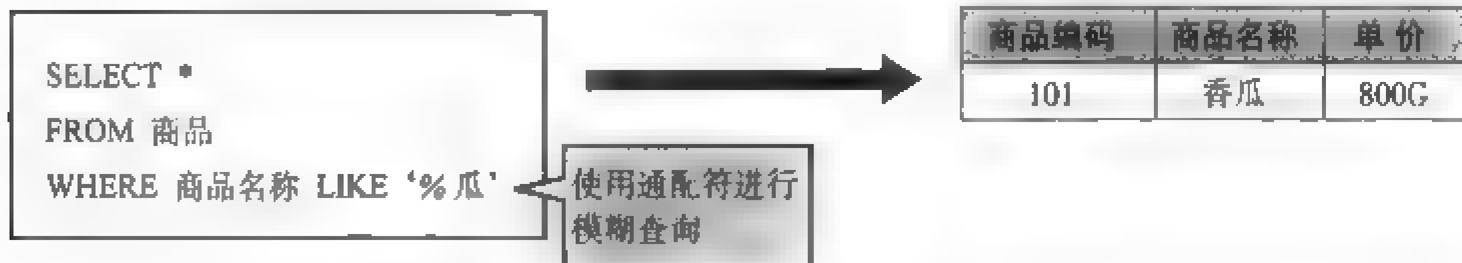
■ 逻辑运算符

逻辑运算符	含义	示例	示例的含义
AND	A 和 B A 或 B	商品编码 > = 200 AND 单价 = 100	商品编码为 200 以上且 单价为 100G
OR	A 或 B	商品编码 > = 200 OR 单价 = 100	商品编码为 200 以上或 单价为 100G
NOT	非 A	NOT 单价 = 100	单价不为 100G

使用通配符设定条件

另外，在设定条件时，还能够进行模糊检索。此时，要用到“%”和“_”等字符来做成通配符，使用 LIKE 检索匹配通配符的字符串。使用可以表示任意字符串的“%”和表示一个字符的“_”，就能够检索与指定部分一致的字符串。

制作通配符时使用的这些字符我们称之为“通配符”(wild card)。请看如下查询。

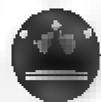


这里我们检索的是商品名称末尾为“瓜”的字符串。模糊部分由“%”这个通配符来代替进行检索。

SQL 命令中使用的通配符有以下几种。

■ 通配符

通配符	含义	示例	匹配的字符串
%	代表任意多个字符	%果 菠%	苹果　无花果 菠萝　菠萝蜜
_	代表任意一个字符	子 香	李子 香瓜



能够进行各种各样的检索

还有各种各样的其他检索方式。例如，在设定数值范围时，可以使用“BETWEEN…AND…”这种语句。通过如下设定，可以调取单价在150G以上200G以下的商品。

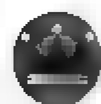
```
SELECT *
FROM 商品
WHERE 单价
BETWEEN 150 AND 200
```

能够设定范围检索

另外，检索含有空值的行时，可以使用“IS NULL”语句。通过如下设定，可以调取单价为空值的商品。

```
SELECT *
FROM 商品
WHERE 单价 IS NULL
```

能够检索空值



设定条件的问题

接下来，我们试着使用各种各样的条件写SQL命令。我们使用下图中的“出口国”表做练习(人口单位：万人)。

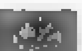
出口国表

出口国表	出口国名称	人口
12	米纳米王国	100
23	阿尔法帝国	120
25	理陀儿王国	150
32	萨藏纳王国	80

使用 SQL 命令回答 Q1~Q6 的问题。

调查人口在 100 万以上的国家，请抽取如下表格。

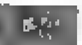
出口国编码	出口国名称	人口
12	米纳米王国	100
23	阿尔法帝国	120
25	理陀儿王国	150



调查人口不足 100 万的国家，请抽取如下表格。

出口国编码	出口国名称	人口
32	萨藏纳王国	80

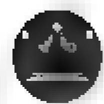
请调查出口国编码不足 20 且人口在 100 万人以上的国家。



请同时调查出口国编码在 30 以上，人口超过 100 万的国家。

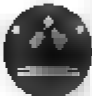
理陀儿王国的人口是多少万人?

请调查国名中含有“米%米”的国家。

 使用计算函数计算

露娜公主和凯恩也了解了各种各样的计算函数。计算函数又称作“集合函数”(aggregate function)。使用计算函数能够统计计算数据。能够进行最大值、最小值、行数、合计等计算。

计算函数和 WHERE 命令同时使用，可以求出限定行的统计值。例如，根据下列指令，可以调查单价在 200G 以上的商品的行数。

 分组计算

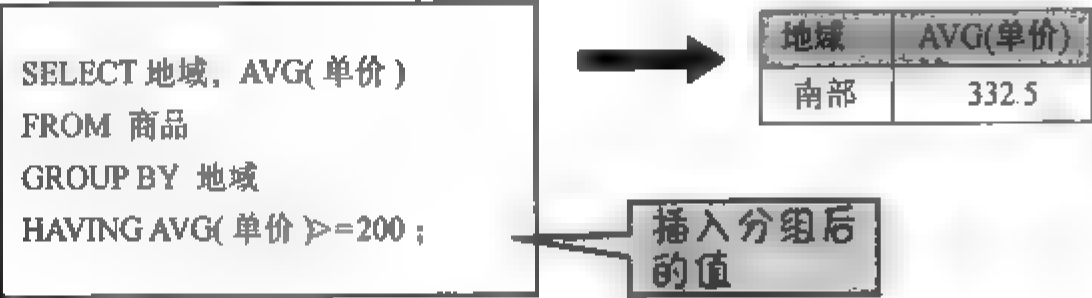
将数据分组化，可以求每一组的统计值。例如，想求每个地域的商品数和平均单价，就可以使用分组化功能。

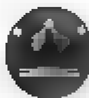
在分组时，使用计算函数和“GROUP BY ”命令组合。如下商品表所示。

“商品”表

商品编码	商品名称	单价	地域
101	香瓜	800G	南部
102	草莓	150G	中部
103	苹果	120G	北部
104	柠檬	200G	南部
201	栗子	100G	北部
202	柿子	160G	中部
301	桃子	130G	南部
302	猕猴桃	200G	南部

关于分组后求得的统计值，如果还要进一步设定条件时该怎么办呢？例如，需要调查每个地域平均单价在 200G 以上的商品。这时，不是使用 WHERE 命令设定条件，而是使用 HAVING 命令来设定条件。这样就只抽取了平均单价在 200G 以上的地域。



 计算与分组化的问题

接下来使用下面的“出口国”表，完成 Q7 至 Q16 中的问题。(人口单位 ：万人)

“出口国”表

出口国编码	出口国名称	人口	地域
12	米纳米王国	100	南洋
15	帕罗努国	200	中部
22	托康国	160	北洋
23	阿尔法帝国	120	北洋
25	理陀儿王国	150	南洋
31	塔哈鲁王国	240	北洋
32	萨藏纳王国	80	南洋
33	马里昂国	300	中部

人口最少的国家有多少万人？

人口最多的国家有多少万人？

出口国中所有国家人口总和是多少万人？

出口国编码为 20 以上的所有国家人口总和是多少万人?

人口在 100 万以上的国家有几个?

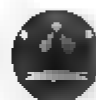
地处北洋的国家有几个?

地处北洋的国家中人口最多的国家有多少万人?

除理陀儿王国外，其他国家人口总和是多少万人?

请调查平均人口在 200 万人以上的地域。

请调查拥有三个国家以上的地域。



使用子查询检索

SQL 中还有更加复杂的查询。在查询中还可以嵌套其他查询。这种查询就称作子查询(Subquery)。请看下表。

“商品”表

商品编码	商品	名称单价
101	香瓜	800G
102	草莓	150G
103	苹果	120G
104	柠檬	200G

“销售明细”表

报表编码	商品编码	数量
1101	101	1 100
1101	102	300
1102	103	1 700
1103	104	500
1104	101	2 500
1105	103	2 000
1105	104	700

使用这两张表，检索数量在 1000 个以上的商品名称时，可以使用如下 SQL 命令进行检索。

```
SELECT * FROM 商品
WHERE 商品编码 IN
(SELECT 商品编码
 FROM 销售明细
WHERE 数量 > = 1000);
```

子查询

此 SQL 命令首先检索“销售明细”上的商品编码，得到的结果为 101 和 103。此商品编码也是另外一个 SELECT 命令中条件的一部分。IN 命令是指与括号中的任意值一致时，条件即成立。因此，商品编码为 101 和 103 的商品即为要检索的商品。

子查询中括号内 SELECT 命令的结果，再由括号外的 SELECT 命令进行检索。内部的检索结果传递给外部检索。因此，根据此子查询命令，得到如下结果：

商品编码	商品名称	单价
101	香瓜	800G
103	苹果	120G

使用相关子查询进行检索

在子查询中，内部查询也可以使用外部指定的表格。这叫做相关子查询 (correlated subquery)。

在如下查询命令中，外部的查询命令中的“销售明细”表也用在了内部的查询命令中。外部设定的表格另起名字，使用在内部的查询命令中。在这里，U 就是“销售明细”表格的别名。

在相关子查询中，因为在内部的查询命令中使用了外部的查询表格，所以仅使用内部的查询命令是无法查询的。

```
   SELECT *
①  FROM 销售明细 U
③  WHERE 数量 >
   (SELECT AVG ( 数量 )
②  FROM 销售明细
   WHERE 商品编码 = U. 商品编码 );
```

报表编码	商品编码	数量
1104	101	2 500
1105	103	2 000
1105	104	700

抽出比每个符合条件的商品大于平均数量的商品明细。

这个相关子查询是如何处理的呢？我们按顺序来看看。在相关子查询中，首先进行外部的查询。

```
①  SELECT *
    FROM 销售明细
```

将此结果的每一行转入内部查询进行评估。也就是说，将已经另起名称为 U 的销售明细的第一行转入内部查询，抽取商品编码相同的行。

```
    (SELECT AVG ( 数量 )
②  FROM 销售明细
    WHERE 商品编码 = 104)
```

例如，第一行商品编码为 101，所以②中平均销售数量为 1800。求得的结果转入外部查询作为条件。

```
③  WHERE 数量 > (1,800)
```

在这个查询当中，抽取“比每个符合条件的商品的平均销售数量高的情况”。这样按②、③步检索①的每一行。因此，在所有数据中仅抽取了①的第 5、6、7 行。

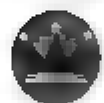
也就是说，此相关子查询求取销售明细表中每个商品的平均销售数量，抽取那些比平均销售数量多的明细项目。

接下来请回答对“商品表”“销售明细”的提问。

调查单价在300G以上的水果的销售明细，抽取如下表格。

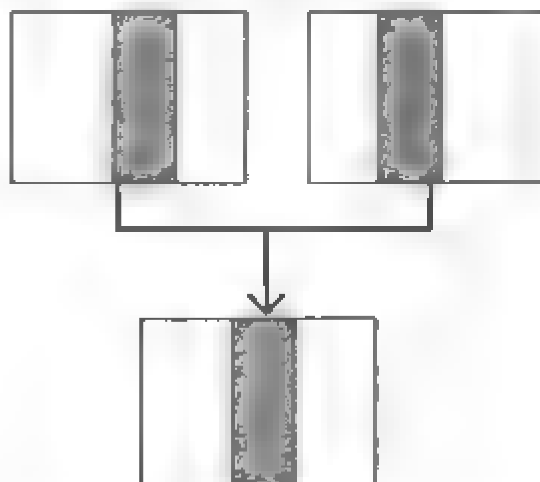
报表编码	商品编码	数量
1101	101	1100
1104	101	2500

求每种商品的平均销售数量，调查低于此平均销售数量的明细项目。



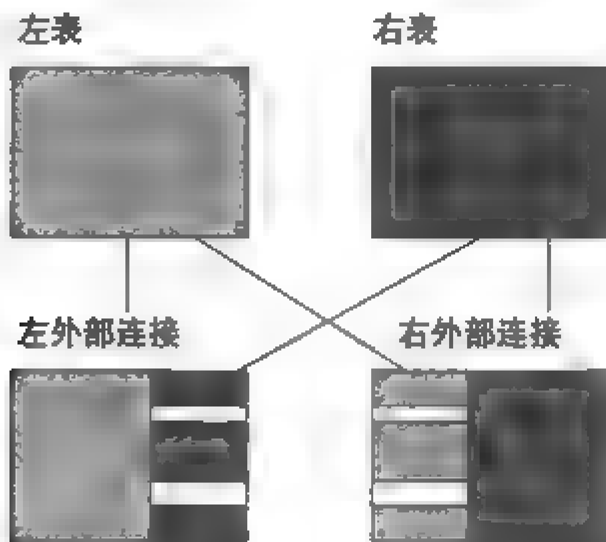
各种各样的连接方法

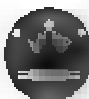
露娜公主和凯恩通过捆绑SQL检索，制成了连接表格的销售报表。像销售报表这样，以相同意义的列为媒介，连接成表格，称作同等连接(equi join)。在同等连接中，设定拥有相同数值的行为连接条件，结合成新的表格。我们称将重复的列整合成一列的连接叫做自然连接(natural join)。一般连接表格时，使用自然连接。



像同等连接这样，仅选择数值相同的行进行连接，我们称之为内部连接(inner join)。

与此相对应的是，全部保留任何一方表格的所有行，将一方中没有的行设置为空值的连接方法称为外部连接 (outer join)。两个表格在 SQL 命令中分左右时，根据保留全部表格行的情况，分为左外部连接 (left outer join) 和右外部连接 (right outer join)。



 制作表格

最后，露娜公主和凯恩学习了制作表格的功能。在制作表格时，使用 CREATE TABLE 命令。由于 CREATE TABLE 命令的内部记述方法因数据库产品的不同而不同，所以在此仅举一例进行说明。

```
CREATE TABLE 商品
(
  商品编码  NUMBER (3, 0),
  商品名称  CHAR (20),
  单价      NUMBER (10, 0),
  PRIMARY  KEY ( 商品编码 )
);
```

制作表格

制作表格时，设定列名。另外对于列，可以设定主键和外键。这里我们设定商品编码为主键 (PRIMARY KEY)。制作表格时，也可以做如下设定。

限制	含义
PRIMARY KEY	设定主关键词
UNIQUE	唯一
NOT NULL	不许出现 NULL 值
CHECK	检查范围
DEFAULT	设定默认值
FOREIGN KEY / REFERENCES	设定外关键词

这些设定称为约束 (constraint)。设定表格时，通过给予限制，防止向表格中输入相互矛盾的数据。这样也就可以正确地管理数据库了。

插入、更新、删除数据

在向使用 CREATE TABLE 命令制作的表格中插入、更新、删除数据时，使用 INSERT 命令、UPDATE 命令和 DELETE 命令。让我们试着使用 SQL 来插入、更新、删除数据吧。

```
INSERT INTO 商品(商品编码,商品名称,单价)
VALUES (202, "枇杷", 200);
```

添加"枇杷"

```
UPDATE 商品
SET 商品名称 = "甜瓜"
WHERE 商品名称 = "香瓜";
```

将"香瓜"变为"甜瓜"

```
DELETE FROM 商品
WHERE 商品名称 = "苹果";
```

删除"苹果"

商品编码	商品名称	单价
101	甜瓜	800G
102	草莓	150G
103	苹果	120G
104	柠檬	200G
202	枇杷	200G

变为“甜瓜”

“苹果”被删除

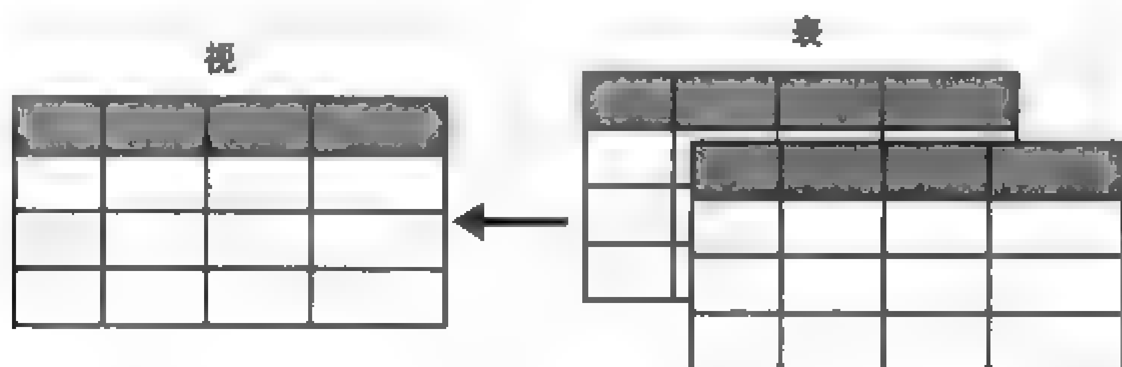
添加了“枇杷”

插入、更新、删除行时，不能违反在 CREATE TABLE 命令中设定的约束。例如在作为主键的“商品编码”列中已经登记了商品编码为“202”的商品时，便不可以再添加“枇杷”了。主键中不能添加重复的数据。

数据的插入、更新、删除操作必须遵循约束条件。

制作视图

基于制成的表格，也可以制作仅从用户的视角看到的、虚拟的表格。这种表格叫做视图（viewed table）。相对于视图来说，输入数据的表格称为基本表 (Base table)。



制作视图时使用如下 SQL 命令：

```
CREATE VIEW 高价商品(商品编码, 商品名称, 单价)
AS SELECT *
 FROM 商品
WHERE 单价>=200;
```

制作视图

“高价商品”表是以作为基本表的商品表制作而成的视图。从“商品”表中抽取价格在200G以上的数据制作而成。

“高价商品”表

商品编码	商品名称	单价
101	香瓜	800G
104	柠檬	200G
202	枇杷	200G

制成视表图后，可以在“高价商品表”中使用基本表中使用的检索。

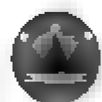
```
SELECT *
FROM 高价商品
WHERE 单价 > = 500 ;
```

视图可以像基本表一样使用

在只希望公开基本表中的一部分数据时，使用视图将非常方便。
另外，还有删除基本表和视图的 SQL 命令。

```
DROP VIEW 高价商品 ;
```

```
DROP TABLE 商品 ;
```



管理表格和数据中的问题

接下来是最后的练习问题。请写出 Q19 至 Q22 的 SQL 命令。(人口单位：万人)

使用 CREATE TABLE 制作“出口国”表。添加以下四条数据。

“出口国”表

出口国编码	出口国名称	人口	地域
12	米纳米王国	100	南洋
15	帕罗努国	200	中部
22	托康塔国	160	北洋
23	阿尔法帝国	120	北洋

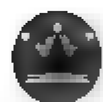
从 Q19 的“出口国”表中，就地处北洋的国家，制作“北洋诸国”视图。

“北洋诸国”表

出口国编码	出口国名称	人口
22	托康塔国	160
23	阿尔法帝国	120

将“出口国”表中托康塔国的人口更改为 150 万人。

将“出口国”表中帕罗努国的数据删除。



从应用程序中使用SQL

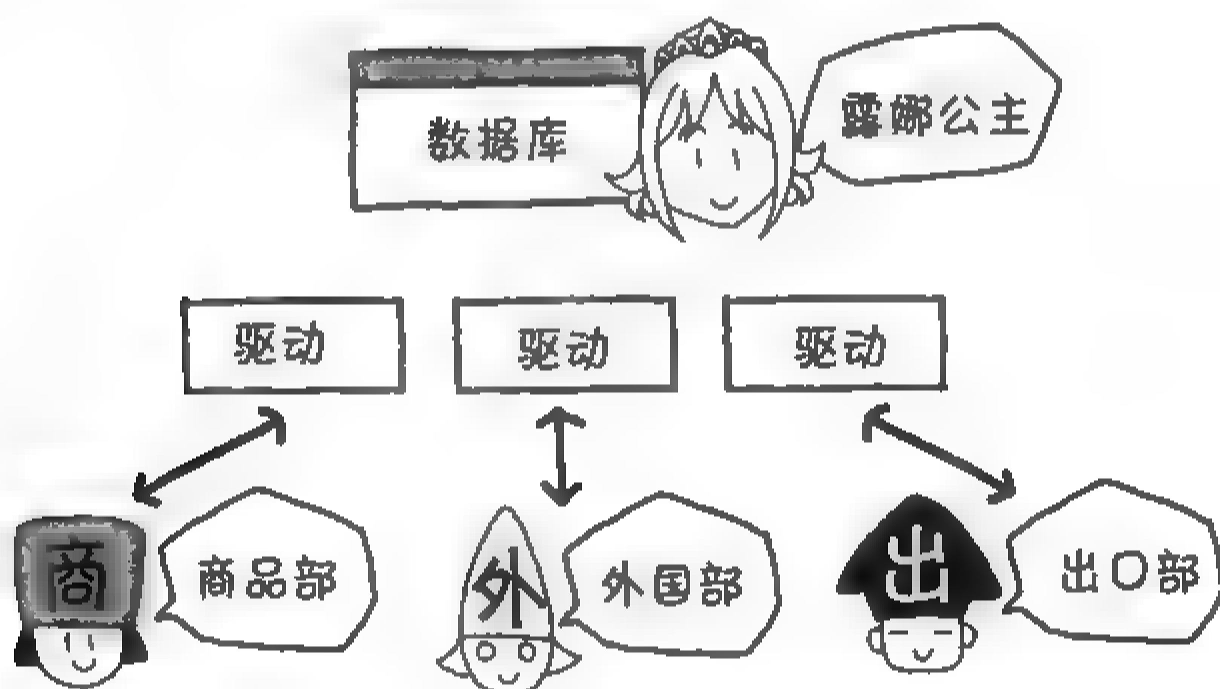
在此之前我们介绍了 SQL 的主要功能。

目前所介绍的 SQL 命令中，每一条命令都是以与数据库对话的形式进行检索的。下面的画面，是使用数据库产品 (MySQL) 附带的对话画面，发出每一条 SQL 命令，获得查询结果的样式。就这样用 SQL 命令与数据库之间进行对话式的查询。

但是，实际使用数据库时，制作销售管理、商品管理等应用程序，在这些程序中使用 SQL 的情况比较多。

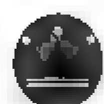
在程序语言使用 SQL 的方式有很多种。编译程序时预置 SQL 命令的静态 SQL 及执行程序时送出 SQL 命令的动态 SQL 等。

例如在使用程序语言 JAVA 时，通过准备动态 SQL 相应的驱动 (driver)，如图所示可以访问、查询数据库。



■ 使用SQL的JAVA编码示例

```
import java.sql.*;

public class Fruits
{
  public static void main(String args[])
  {
     try{
       String drv = "com.mysql.jdbc.Driver";
       String url = "jdbc:mysql:/// fruitsdb";
       String usr = "";
       String pw = "";

       Class.forName(drv);
       Connection cn = DriverManager.getConnection(url, usr, pw);

       Statement st = cn.createStatement();
       String qry = "SELECT * FROM fruits";

       ResultSet rs = st.executeQuery(qry);      使用SQL的JAVA编码示例

       ResultSetMetaData rm = rs.getMetaData();
       int cnum = rm.getColumnCount();

       while(rs.next()){      执行时发送的SQL命令
          for(int i=1; i<=cnum; i++){
              System.out.print(rm.getColumnName(i) +  "--"+ rs getObject(i) + "  ");
          }
          System.out.println("");
       }
...
}
```



使用游标移动行

到目前为止，用对话的形式使用 SQL 的情况中，可以将查询结果做成一个表集中处理。但是，一般的程序语言中，没有使用一条命令集中处理多个行的功能。因此，在程序语言中使用 SQL，作为抽取结果的表格，访问每一行的方法非常必要。

此时，我们可以使用游标这个显示行位置的概念。可以通过游标换行，访问每一行。使用程序语言的循环语句 (loop statement) 移动游标，可以访问多个行。使用游标抽取每行的数据，称作提取 (fetch)。JAVA 编码的列表也可以访问每行数据。

商品编码	商品名称	单价
101	香瓜	800G
102	草莓	150G
103	苹果	120G
104	柠檬	200G

这样处理后，可以使应用程序对数据库发送 SQL 命令，从而进行查询。

小结

- SQL 具有数据定义功能、数据操作功能和数据约束功能。
- 检索数据时使用 SELECT 命令。
- 指定条件时使用 WHERE 命令。
- 插入、更新、删除数据时使用 INSERT、UPDATE 和 DELETE 命令。
- 制作表格时使用 CREATE TABLE 命令。

答 案

A1

```
SELECT *
FROM 出口国
WHERE 人口 > = 100;
```

A2

```
SELECT *
FROM 出口国
WHERE 人口 < 100
```

A3

```
SELECT *
FROM 出口国
WHERE 出口国编码 < 20 AND 人口 > = 100;
```

出口国编码	出口国名称	人口
12	米纳米王国	100

A4

```
SELECT *
FROM 出口国
WHERE 出口国编码 > = 30 OR 人口 > 100;
```

出口国编码	出口国名称	人口
23	阿尔法帝国	120
25	理陀儿王国	150
32	萨藏纳王国	80

A5

```
SELECT 人口
FROM 出口国
WHERE 出口国名称 ='理陀儿王国';
```

人口
150

A6

```
SELECT *
FROM 出口国
HERE 出口国名称 LIKE'米%米';
```

出口国编码	出口国名称	人口
12	米纳米王国	100

A7

```
SELECT MIN(人口)
FROM 出口国
```

MIN(人口)
80

A8

```
SELECT MAX(人口)
FROM 出口国
```

MAX(人口)
300

A9

SELECT SUM(人口)

FROM 出口国;

SUM(人口)
1 350

A10

SELECT SUM(人口)

FROM 出口国

WHERE 出口国编码 >=20;

SUM(人口)
1 050

A11

SELECT COUNT(*)

FROM 出口国

WHERE 人口 >=100;

COUNT(*)
7

A12

SELECT COUNT(*)

FROM 出口国

WHERE 地域 =' 北洋 ';

COUNT(*)
3

A13

SELECT MAX(人口)

FROM 出口国

WHERE 地域 =' 北洋 ';

MAX (人口)
240

A14

SELECT SUM(人口)

FROM 出口国

WHERE NOT(出口国名称 = ' 理陀儿王国 ');

SUM(人口)
1 200

A15

SELECT 地域 , AVG(人口)

FROM 出口国

GROUP BY 地域

HAVING AVG(人口)>=200;

地域	AVG(人口)
中部	250

A16

SELECT 地域 , COUNT(*)

FROM 出口国

GROUP BY 地域

HAVING COUNT(*)>=3;

地域	COUNT(*)
南洋	3
北洋	3

A17

```
SELECT * FROM 销售明细 WHERE 商品编码 IN
(SELECT 商品编码 FROM 商品 WHERE 单价>=300);
```

报表编码	商品编码	数量
1101	101	1100
1104	101	2500

A18

```
SELECT *
FROM 销售明细 U
WHERE 数量<
(SELECT AVG(数量)
FROM 销售明细
WHERE 商品编码=U.商品编码);
```

报表编码	商品编码	数量
1101	101	1100
1102	103	1700
1103	104	500

A19

```
INSERT INTO 出口国(出口国编码, 出口国名称, 人口, 地域)
VALUES(12, '米纳米王国', 100, '南洋');

INSERT INTO 出口国(出口国编码, 出口国名称, 人口, 地域)
VALUES(15, '帕罗努国', 200, '中部');

INSERT INTO 出口国(出口国编码, 出口国名称, 人口, 地域)
VALUES(22, '托康塔国', 160, '北洋');

INSERT INTO 出口国(出口国编码, 出口国名称, 人口, 地域)
VALUES(23, '阿尔法帝国', 120, '北洋');
```

A20

```
CREATE VIEW 北洋诺国(出口国编码，出口国名称，人口)
AS SELECT 出口国编码，出口国名称，人口
FROM 出口国
WHERE 地域='北洋';
```

A21

```
UPDATE 出口国
SET 人口=150
WHERE 出口国名称='托康塔国';
```

A22

```
DELETE FROM 出口国
WHERE 出口国名称='帕罗努国';
```

SQL 的标准化

SQL 通过了 ISO（国际标准化组织；International Organization for Standardization）的标准认证。在日本也通过了 JIS（日本工业标准；Japanese Industrial Standards）的标准认证。

SQL 标准包括 1992 年颁布的 SQL92 版和 1999 年颁布的 SQL99 版。关系数据库产品可以基于这些标准，进行查询。

但是，关系数据库产品中也有使用自己独有样式的产品。详情可参照数据库产品的手册。

第 5 章

数据库的应用

什么是事务

咔嗒 咔嗒

啪——

咔嗒 咔嗒

咔嗒

咔嗒

咔嗒咔嗒

嗯，
这个和这个要添加……

真是多亏了小T呀！

还有很多很多不明白的地方呀……

例如，
为什么这么多人同时使用数据库都不会出问题呢……

另外，
还有点担心安全的问题……

看来对使用数据库还有很多疑惑呐……

是啊！

对此

嗯嗯

我稍稍研究了一下……

哦？

你的问题……

ど——ん

"为什么这么多人同时使用数据库都不会出问题呢！？"

嗯——

我用简单易懂的图画来解释一下！！

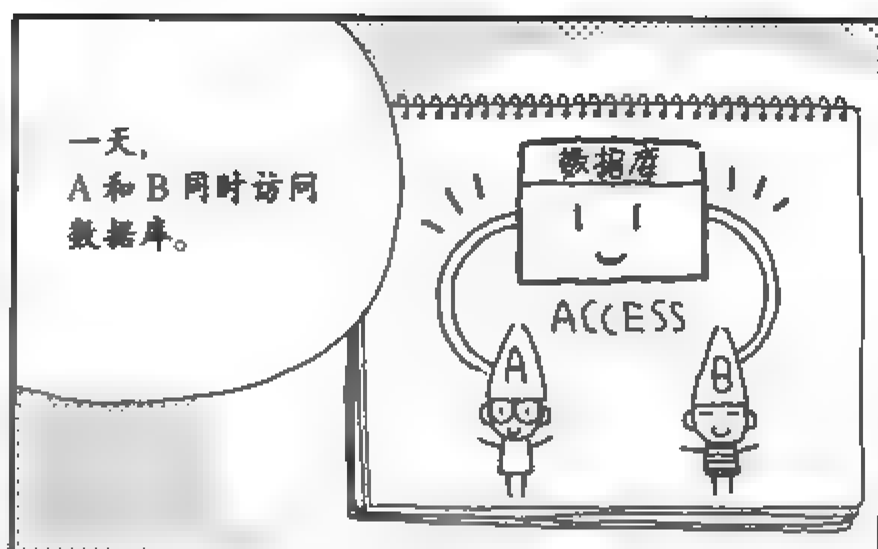
哇，
好厉害

请看！！

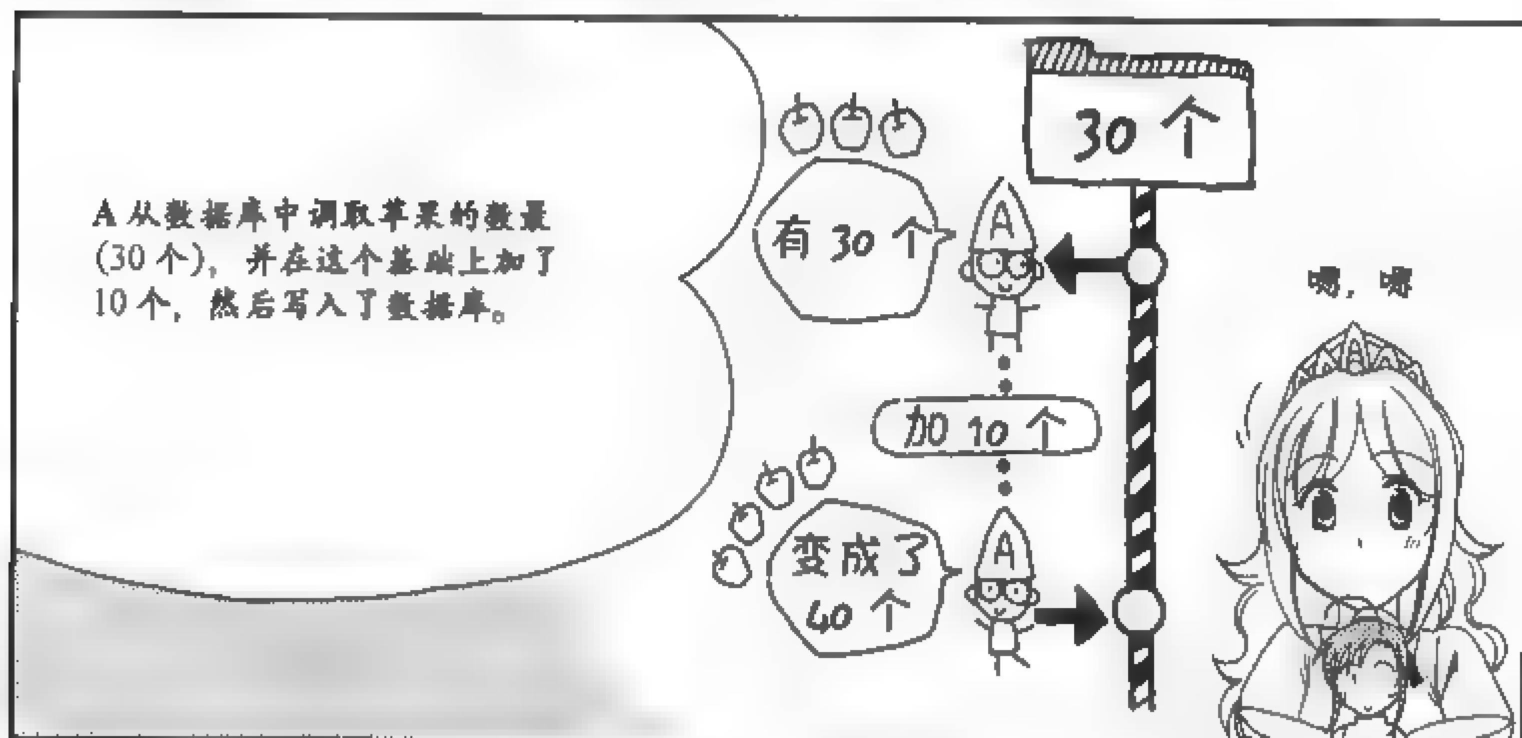
数据库
剧场

干劲儿十足嘛

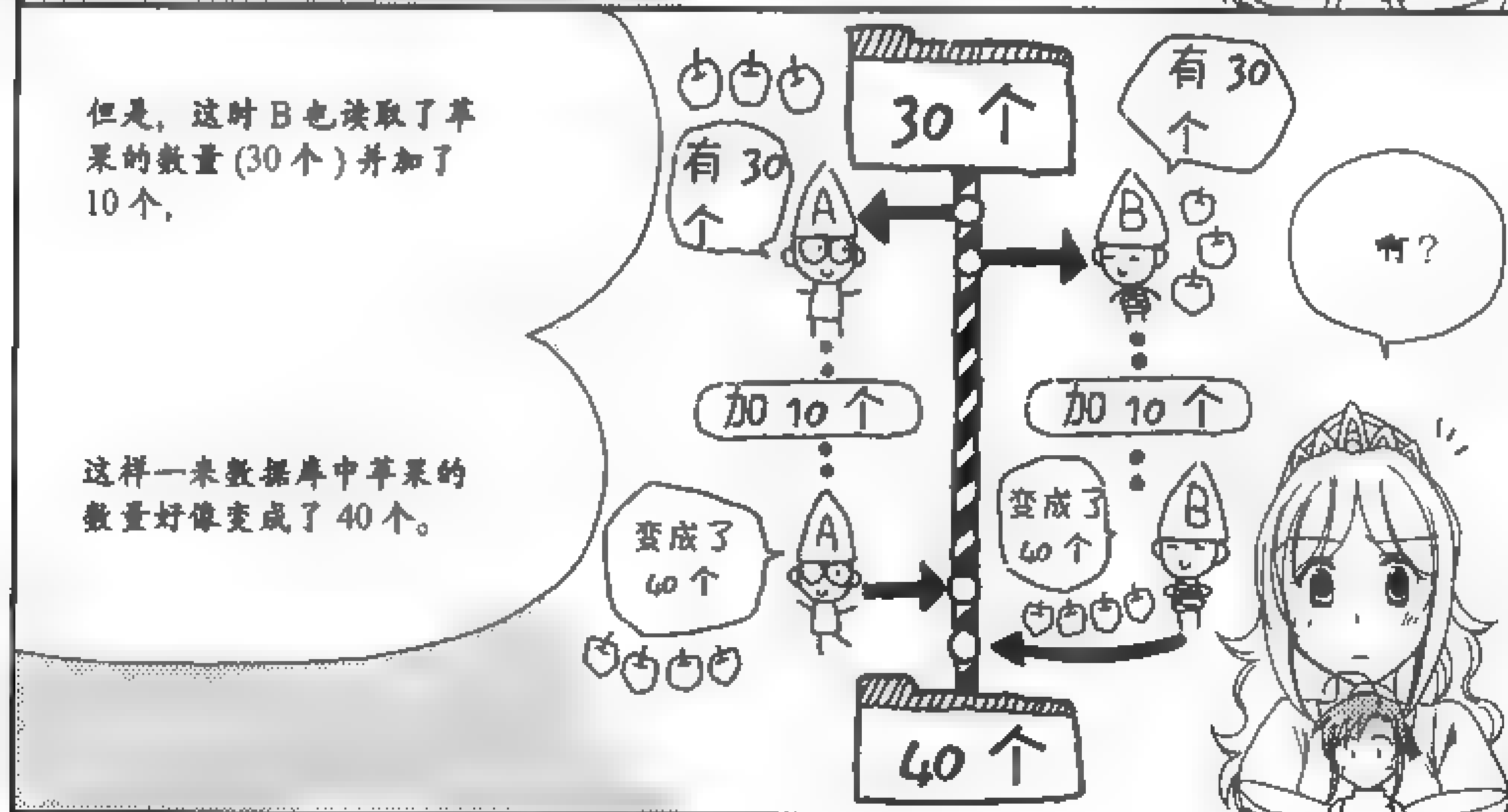
那么，
开始吧！
鼓掌鼓掌
哇



一天，
A和B同时访问
数据库。
数据库
ACCESS
A
B



A从数据库中调取苹果的数量(30个)，并在这个基础上加了10个，然后写入了数据库。
30个
有30个
加10个
变成了40个
嗯，嗯

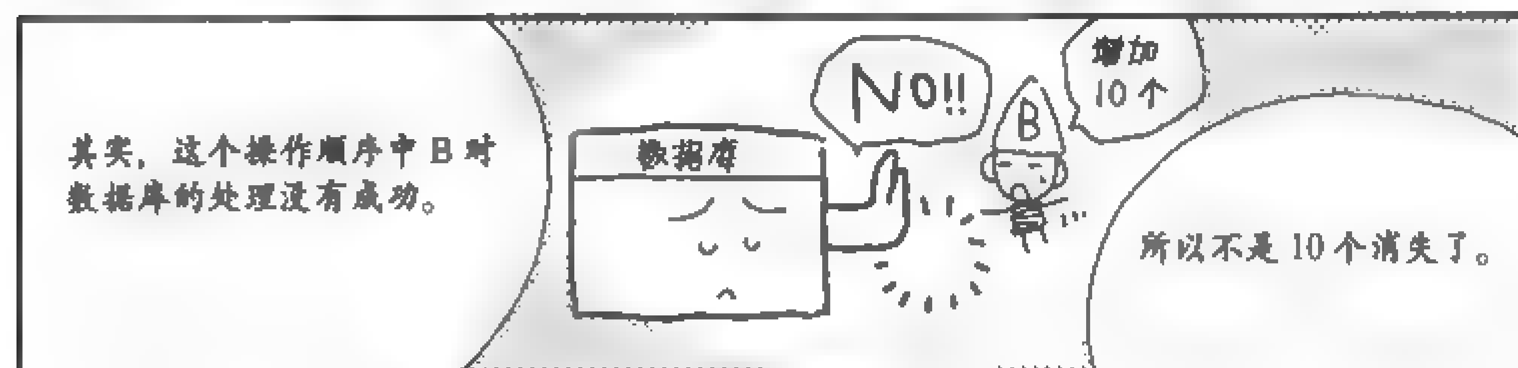


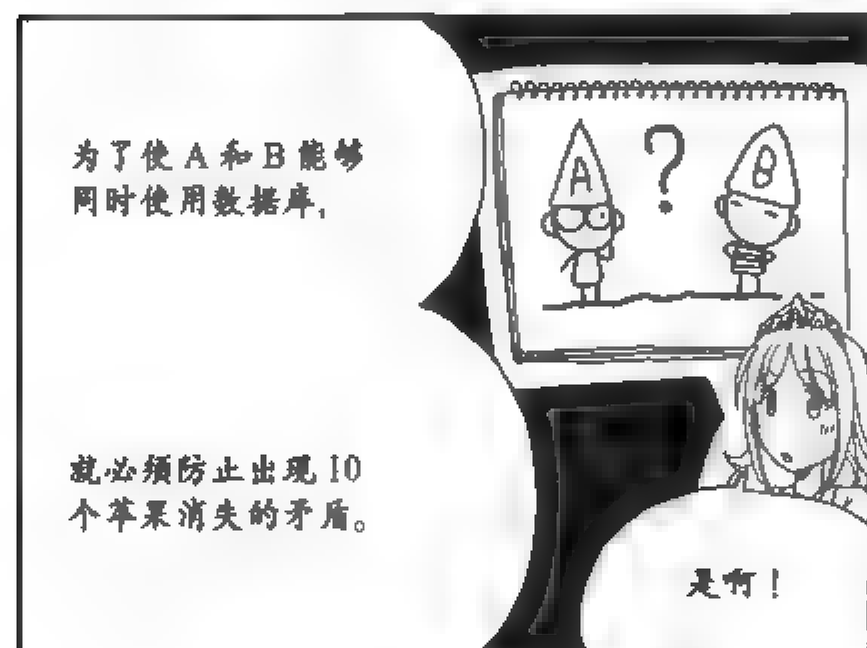
但是，这时B也读取了苹果的数量(30个)并加了10个，
这样一来数据库中苹果的数量好像变成了40个。
30个
有30个
有30个
加10个
加10个
变成了40个
变成了40个
40个
嗯？

不应该是
50个吗？
是这样的！
A加了10个
B加了10个

那么，
那10个哪儿去了？
露娜加油—
嗯——

你吃掉了！
不对！
どーん
どどーん


其实，这个操作顺序中B对
数据库的处理没有成功。
数据库
NO!!
增加
10个
B
所以不是10个消失了。


为了使A和B能够
同时使用数据库，
A
?
B
就必须防止出现10
个苹果消失的矛盾。
是啊！

可是数据库是如何控制
用户的操作的呢？
接下来我们来介绍这个。
凯恩今天好厉害啊~
是啊
来吧！！

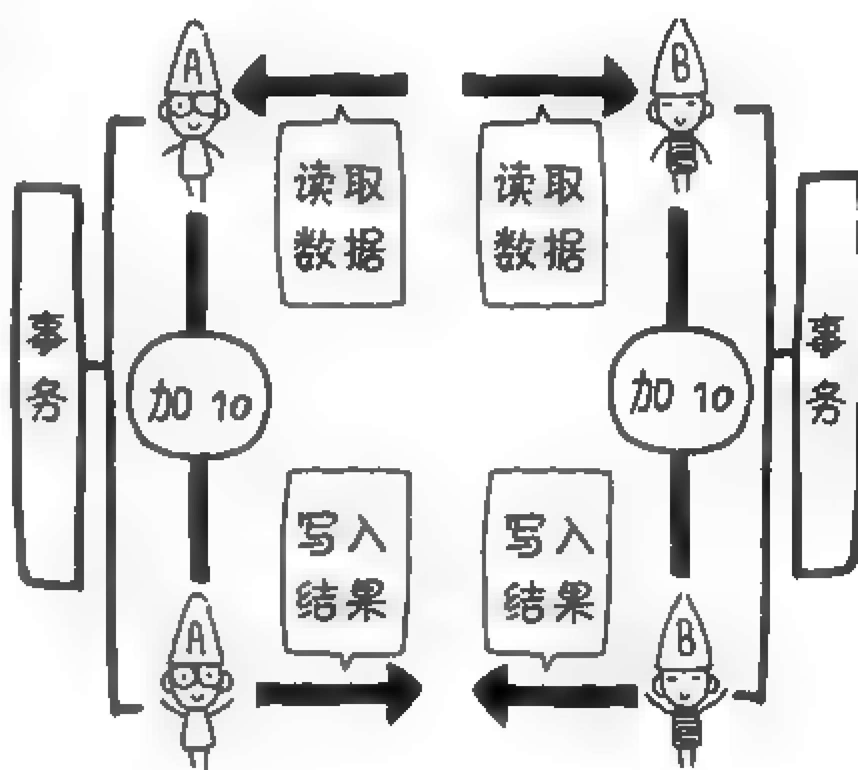
首先，
数据库会将有意义的
数据操做集中处理。

嗯
嗯

这一操作单位我们称之为
“事务”(transaction)。

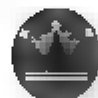
事务？

在这里，
读取数据作业和加法写
入作业就归为一个事务。



也就是说，
A 进行的操作为一
个事务，

B 进行的操作也是
一个事务。



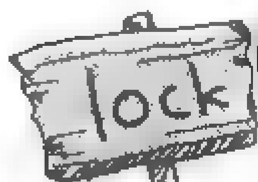
什么是锁

为了使多个用户同时访问数据库时不会发生问题，

自然要控制这些操作。

原来如此！

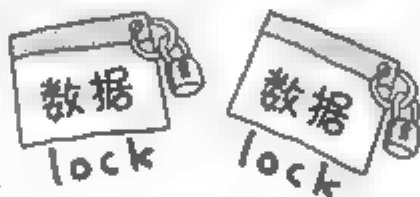
控制的方法我们通常称之为“锁”(LOCK)。



锁？用钥匙开的那种锁吗？

是的！

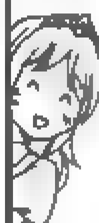
为了防止对重要数据的误操作而上的锁。



哦！

再看之前的那个例子。

好多画儿啊

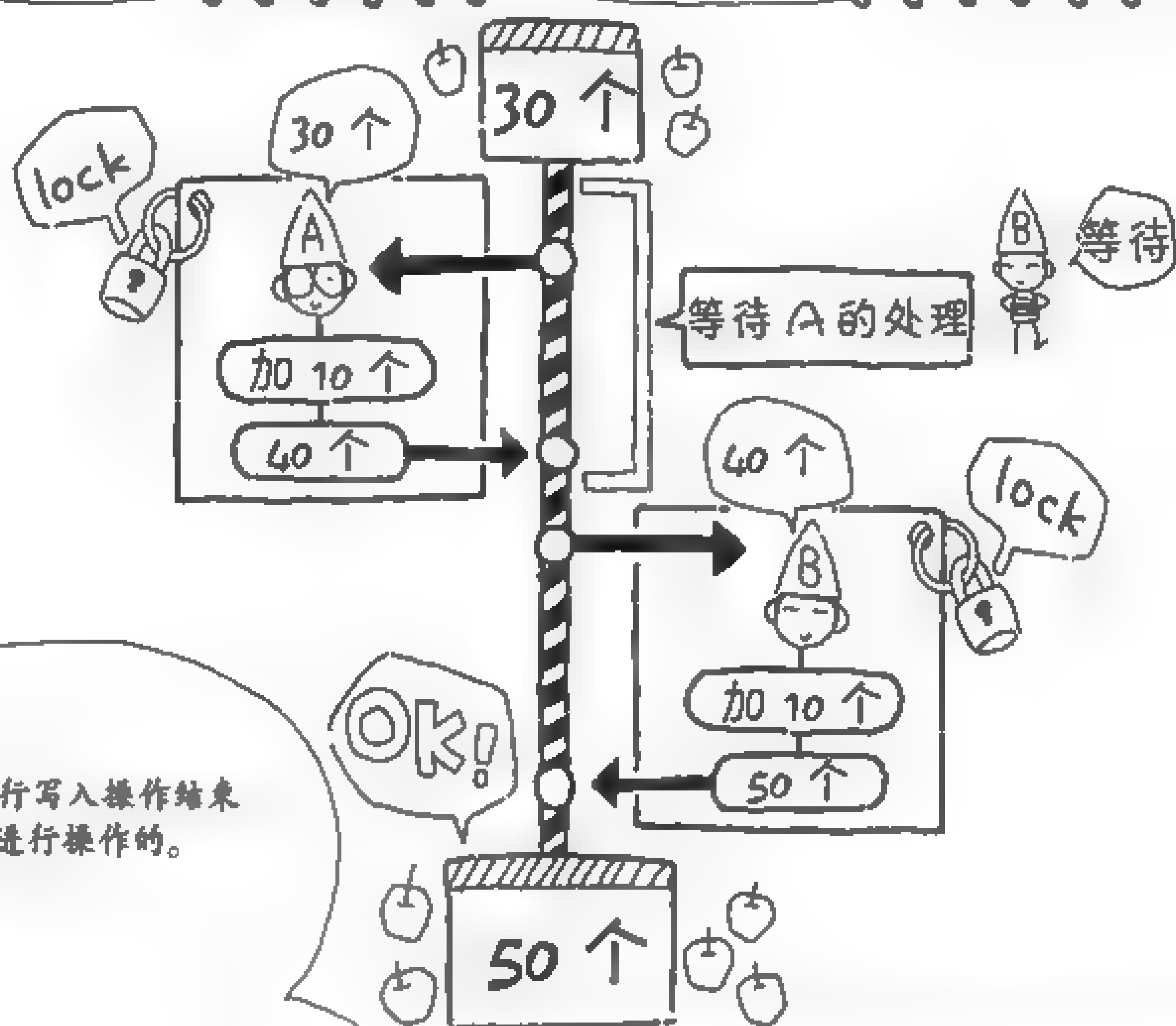


唰

唰

A在进行自己的一系列操作前先上锁的话，

B即使是想进行操作也要等到A的操作完成之后才能开始。



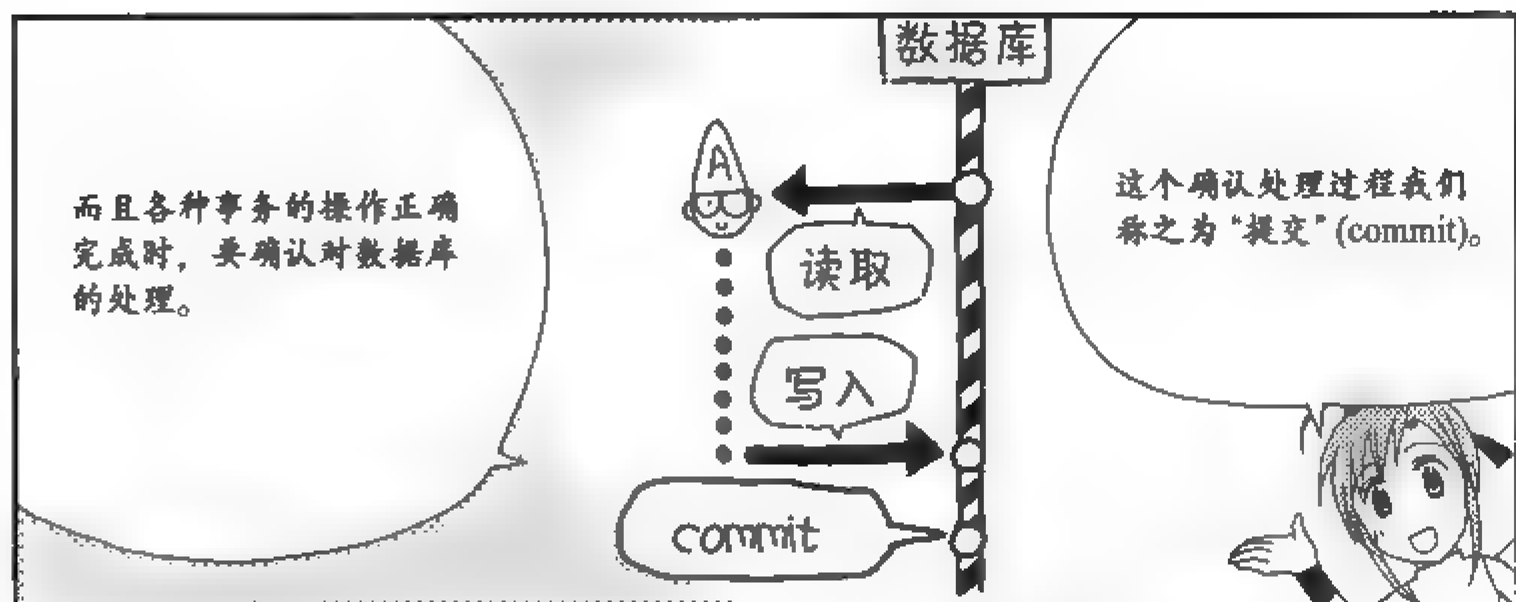
B在A进行写入操作结束前是不能进行操作的。

这样数据库就能给出正确的50个的结果了。

嘿嘿

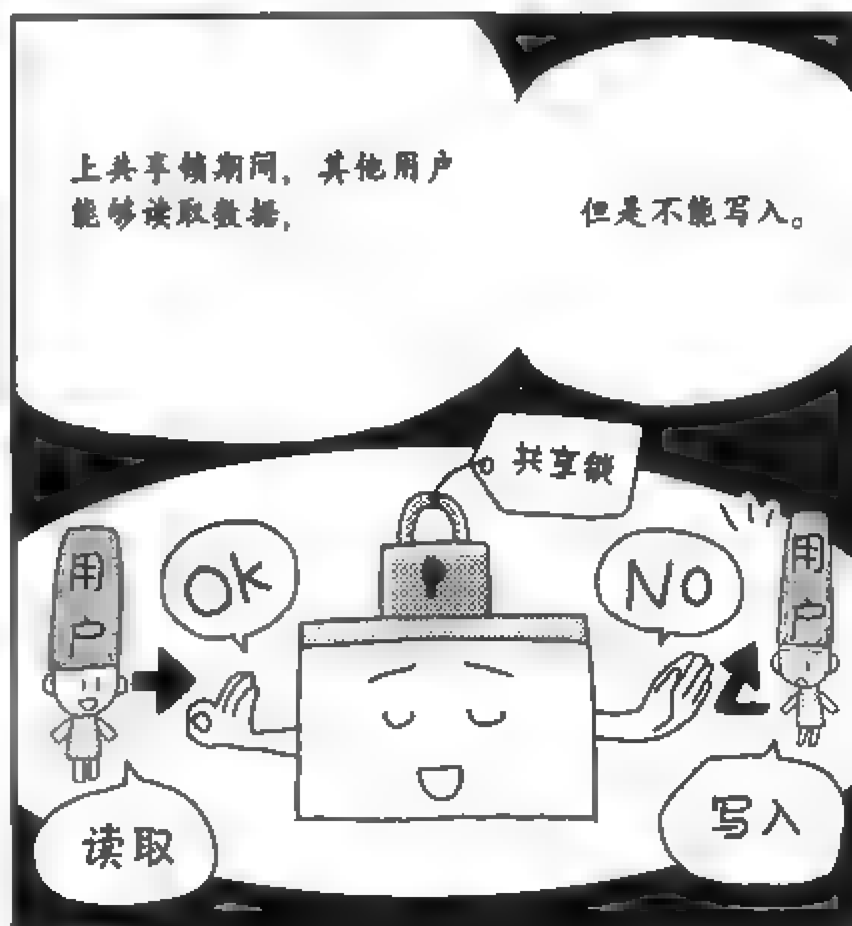
哦……

凯恩好厉害啊


而且各种事务的操作正确完成时，要确认对数据库的处理。
数据库
A
读取
写入
commit
这个确认处理过程我们称之为“提交”(commit)。

通常会根据情况分开使用锁。
不过，虽然使用了锁，但由于数据库是多人共享数据，非常频繁上锁是不太好的。
怎么做呢？

首先用户仅进行读取操作时，使用共享锁。
共享锁


上共享锁期间，其他用户能够读取数据，
但是不能写入。
共享锁
用户
OK
No
用户
读取
写入

用户想要进行写入操作时，就上独占锁。

独占锁

上独占锁时，其他用户既不能读取也不能写入。

这样啊

独占锁

No!!

No!!

用户

用户

读取

写入

锁也有很多种。

使用锁控制多个事务我们称之为同时执行控制 (concurrence control)。

是的！！通过在数据库中使用同时执行控制，

即使再多的人使用也不会发生矛盾。

是吧？凯恩！！

啊……是。

学得相当好呀！
很厉害呦！

是吗?

哈，哪里。

凯恩很不错
哟，露娜。♪

呵呵

是，是吗？

哈，继续，
继续。

啊，好的！

嘣

回到之前
所说的，

即使通过锁来进行同时执行
控制，也有可能发生问题。

A先生

B先生

例如，

A给苹果数据上
了独占锁，

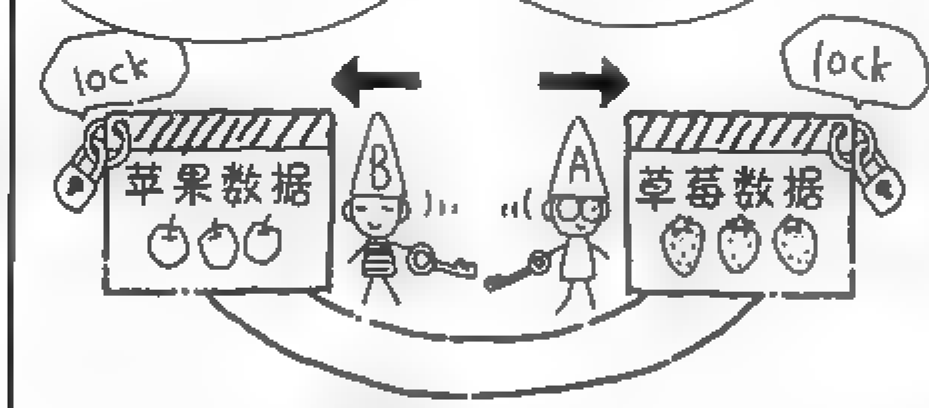
同时B给草莓数据
上了独占锁。

先生

lock

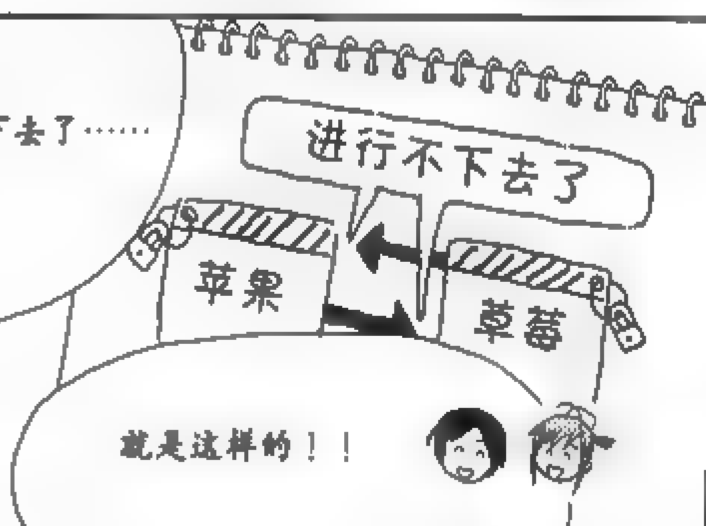
lock

嗯，嗯！


接下来，
A 想给草莓数据上
独占锁，
B 想要给苹果数据
上独占锁。
lock
lock
苹果数据
B
A
草莓数据

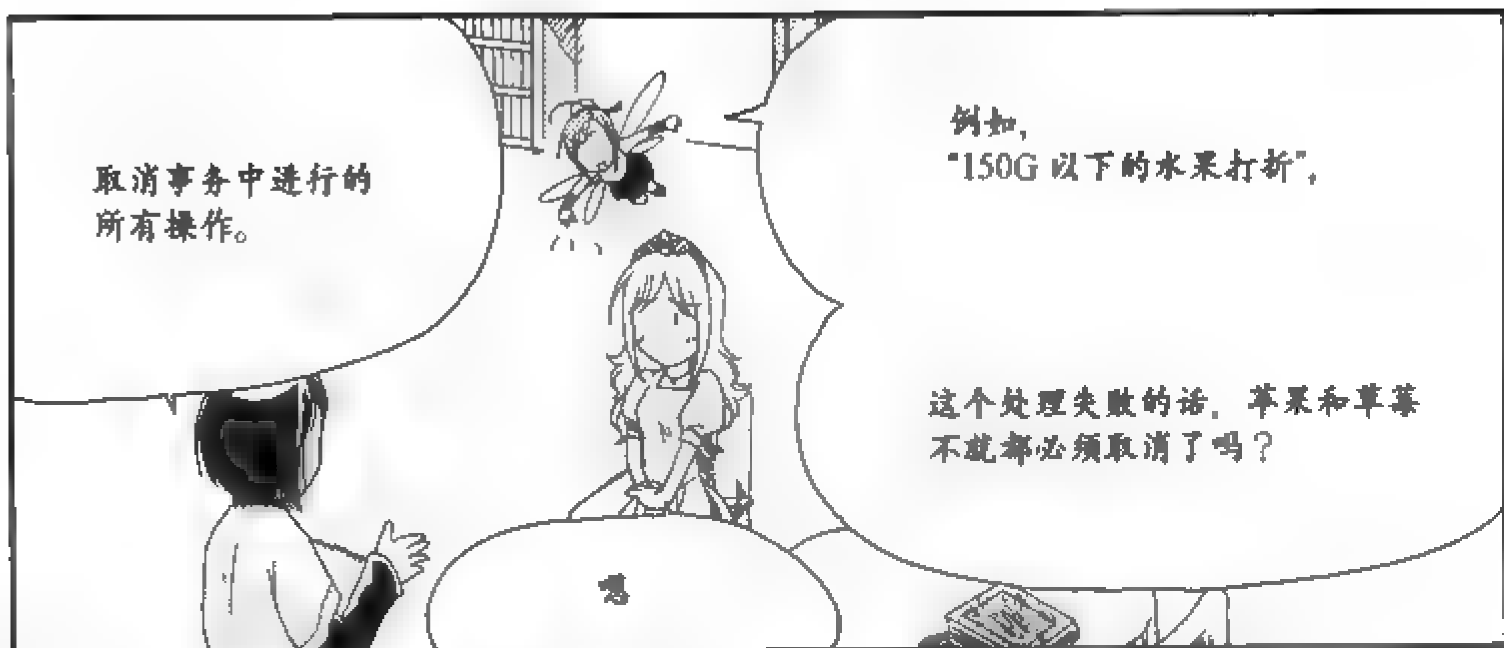
这时会发生
什么情况呢？
咦？

由于任何一方都要等
对方解除锁，
处理就进行不下去了……
是吗？
嗯……
嗯……

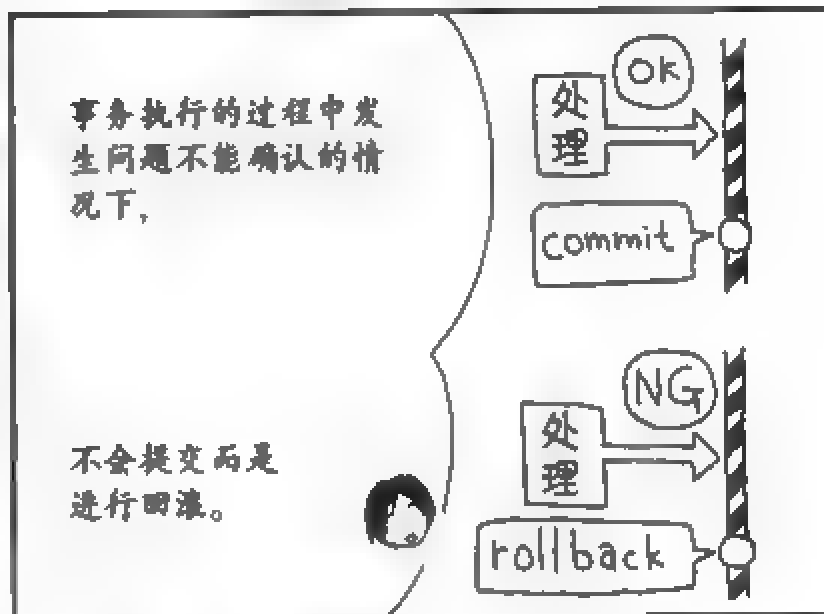

进行不下去了
苹果
草莓
就是这样的！！

为了解决这个问题，
就必须先解除其中一
方的锁。
例如调查持续待机
一定时间的事务。
取消
采取取消的方法！

取消事务我们称之为
回滚 (rollback)。
回滚？

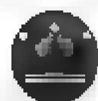

取消事务中进行的所有操作。
例如，“150G以下的水果打折”，
这个处理失败的话，苹果和草莓不就都必须取消了吗？
嗯

从一开始的全部都取消了，是吗？
啊，是的。


事务执行的过程中发生问题不能确认的情况下，
不会提交而是进行回滚。
处理
OK
commit
处理
NG
rollback

是的！事务必须由提交或回滚中的任何一个来结束。
不可能中途结束，是吗？

好！小T，接下来……



数据库的安全问题

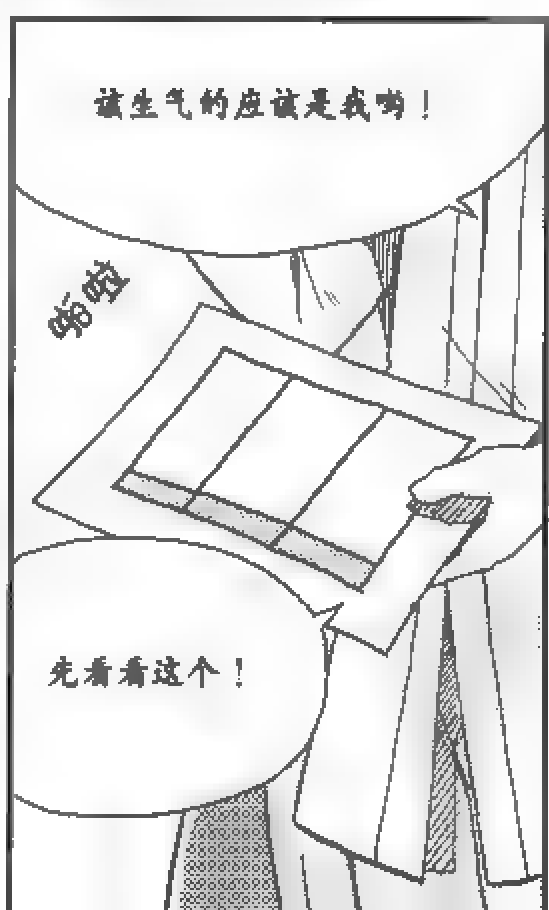
哇!
原来如此……如果众人都好好使用的话，那个数据库就可以避免发生问题了，是吧？
一闪一闪

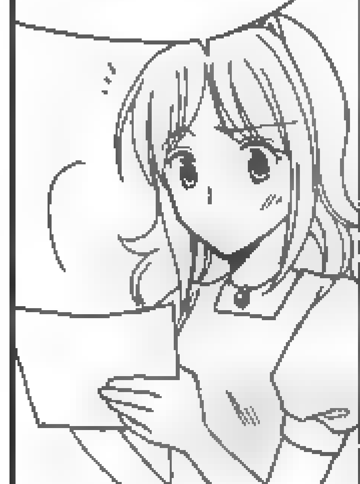
哈哈哈
拉来内斯！！
你什么时候！！
拉来内斯如风而至！

什么呀
HA HA
…

我不是"小T"。

……
你，
你来干什么！！
不要那么生气嘛！


该生气的应该是我哟！
哗啦
先看看这个！

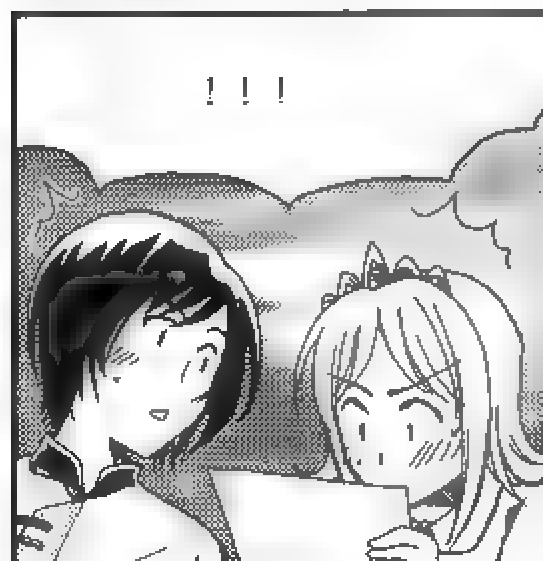


▼商品表

商品编码	商品名称	单价
101	香瓜	10000G
102	草莓	12500G
103	苹果	8000G
104	柠檬	6000G
201	栗子	9000G
202	柿子	12400G
301	桃子	5000G
302	猕猴桃	6000G

这个怎么了？

价格，看看价格！！



唉呀唉呀
请款书上的金额全都错了，
进口国都乱成一锅粥了！！
数据库这东西
可真麻烦呀！
你打算怎么赔偿呢……

ぼっ
露娜公主，嫁到
我国来吧……

忽·视

拉米内斯
殿下！

可能是某个对王国
心怀不轨的人更改
了数据造成的……
为什么会？
还真是无视
我呀！
你们这帮人！

ぼっ
对不起，
请您原谅！

为了不再发生这样的事，我们会好好商量对策的。

……

可是，
这次无论如何…

说到对策了…
到底是怎么回事呢？

问题的原因可能是出在王国中谁都能使用数据库这一点上。

那么，
首先要控制能够访问数据库的人。

唔……

检查一下通过输入用户名和密码访问数据库的人就行了。

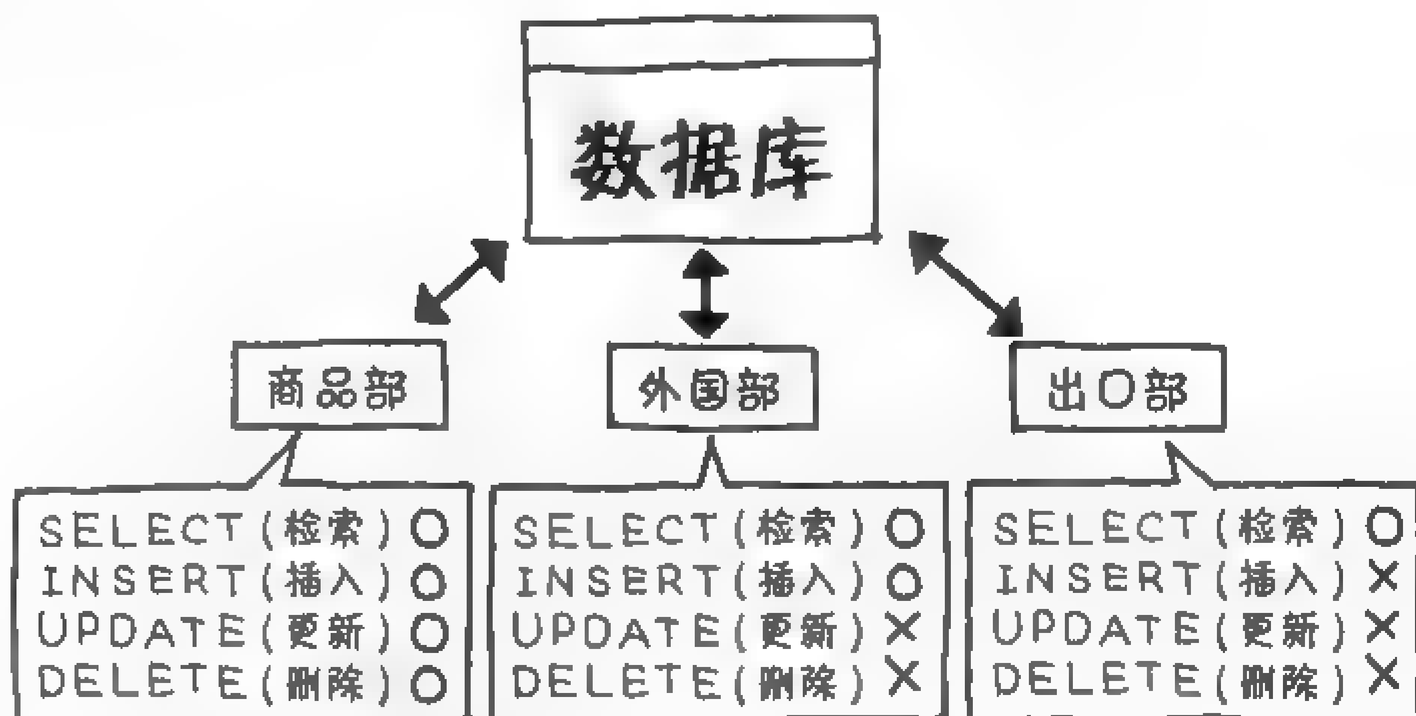
原来如此！

第二，
对用户设定可以做这种操作的权限。

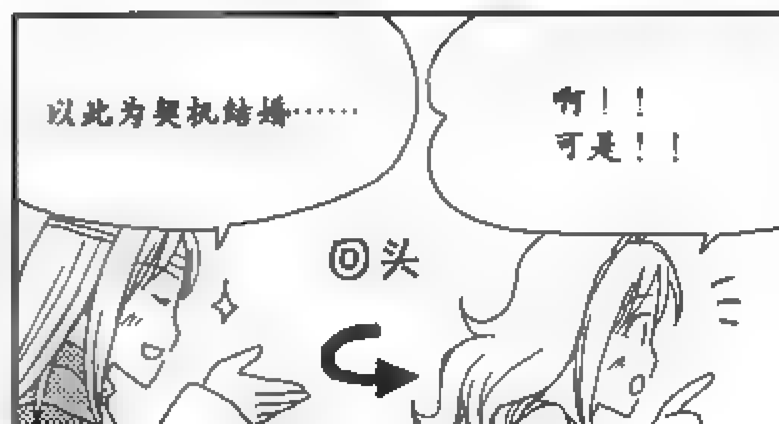
啪!!

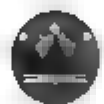
能够对数据进行检索、插入、更新、删除。
能够对数据进行检索、插入，但不能更新、删除。
能够对数据进行检索，但不能插入、更新、删除。

- 商品部的人员能够检索、插入、更新、删除商品数据。
- 外国部的人员能够检索、插入商品数据，但不能更新、删除数据。
- 出口部的人员能够检索商品数据，但不能插入、更新、删除数据。



砰！！





通过索引提高速度

还是
无视
数据库的规模变得更大，
更多的人使用数据库时，
……
可能会在其他地方
出问题……

是啊……
嗯——
例如，
增加数据的话，

检索商品可能会变得
很慢……
是啊！！

数据库不可信吧？
呵呵

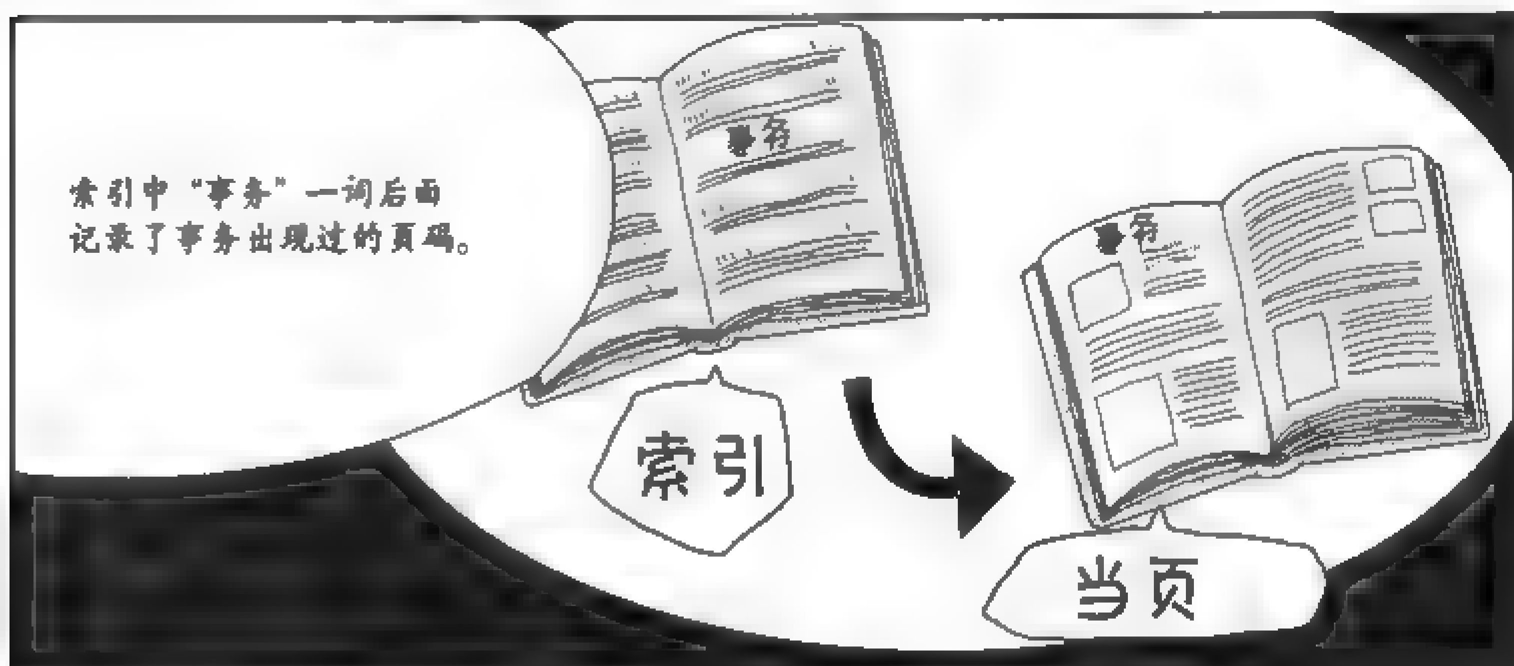
那时……
哈哈

只要设定索引 (index) 就好了。
索引？

看起来好像书后面附
的"索引"？
公主……
结婚更重要啦……
是的！

例如，
想要用这本数据库的书查询"事务"
这个词语的意思，
嗯

就那么一页一页地找的话太麻烦
了，所以看一下书后的索引，
嗯
哼


索引中"事务"一词后面
记录了事务出现过的页码。
事务
索引
事务
当页

使用索引可以很快找到
想找的内容呀！
是的！

数据库也是同样道理。
例如 …
嗯？

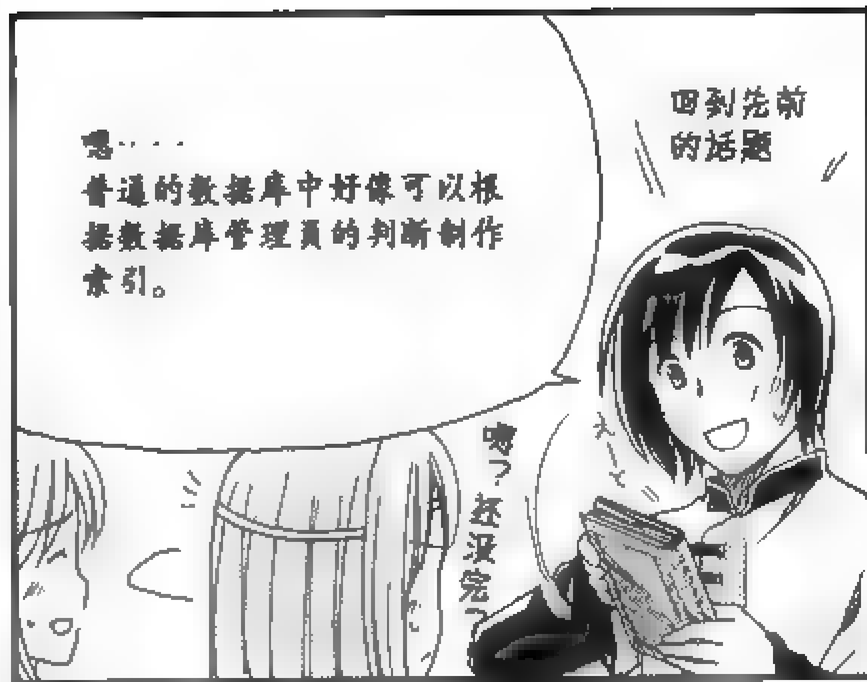
按照商品编码制成索引后
就可以马上知道商品编码是"101"的商品数据的存储位置了。
101 香瓜
数据
也就知道商品存在硬盘的什么地方了。

通过制作索引可以大大地提高检索的速度。
不太明白
怎么……

检索数据时查阅所有行的话太费时间了。
从头开始讲好吧
不用了
ぴょっ

使用索引功能的话，可以减少访问硬盘的次数哟！
减少访问硬盘的次数就可以提高速度了。
はっ
ん？

喂，刚才跟谁……
公主
哈！！
有什么东西吗？

只是自言自语罢了！
哈哈哈，
在这里
拉米内斯看不到呐…

回到先前的话题
嗯…
普通的数据库中好像可以根据数据库管理员的判断制作索引。
嗯？还没完？

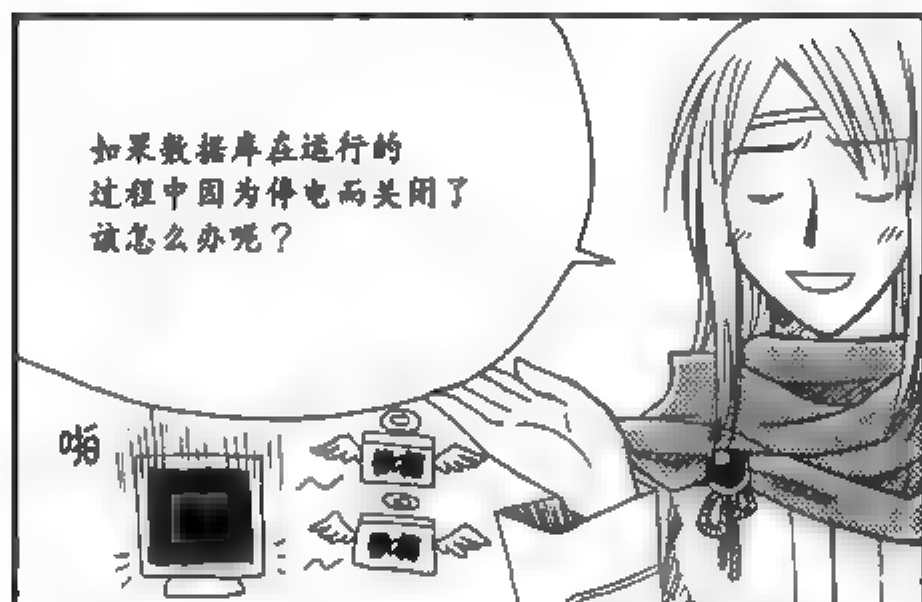
但是因为可以自己制作，如果过分生成索引的话反而会降低效率。
那个？
到底会怎样？
好？还是不好？
悄悄地
啊……

即使是书，
如果索引太长的话，
不就本末倒置了吗？
嗯
而且，
数据频繁更新的话，每次都要更改索引，还是会耽误更新速度的。

嗯，
好像是这样…
哈哈哈哈
是小T说的。

原来如此…
比想象中要好用，
不是吗？
哪儿来
的玫瑰
……
数据库……

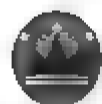
这一点我很佩服！
可是，
还有别的问题哟！！
咚
啊
例如……


如果数据库在运行的
过程中因为停电而关闭了
该怎么办呢？
啪

那个，
没问题哟！

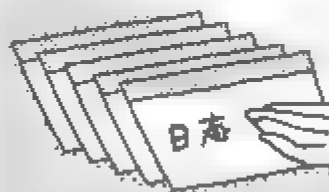
因为数据库有故障
恢复功能。
故障恢复？

也就是说，
即使数据库发生问题也
可以恢复？
原来如此……

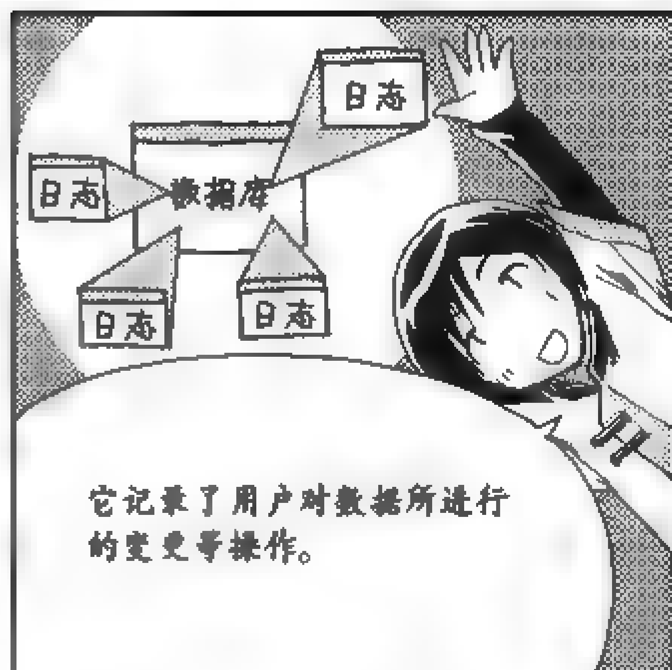


数据库的故障恢复

在数据库中，正在处理数据时会生成一个叫做日志的记录，对吧？
是的！



日志

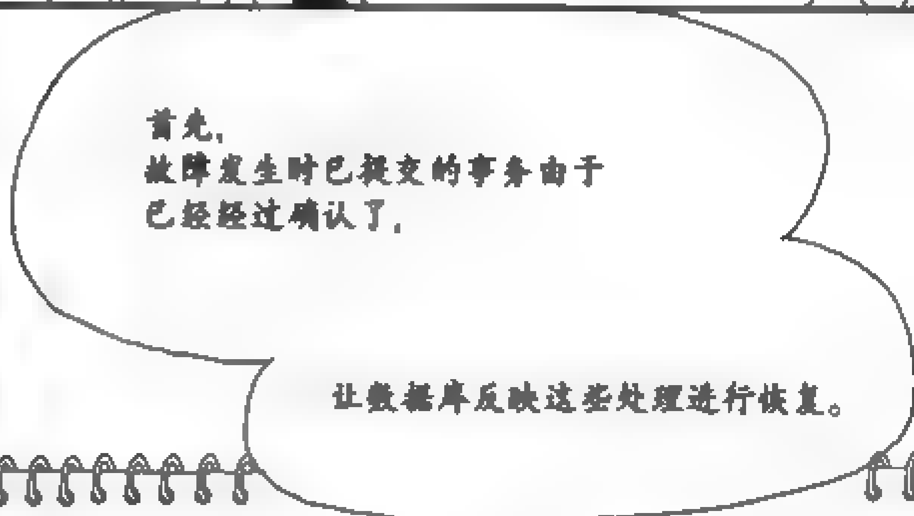


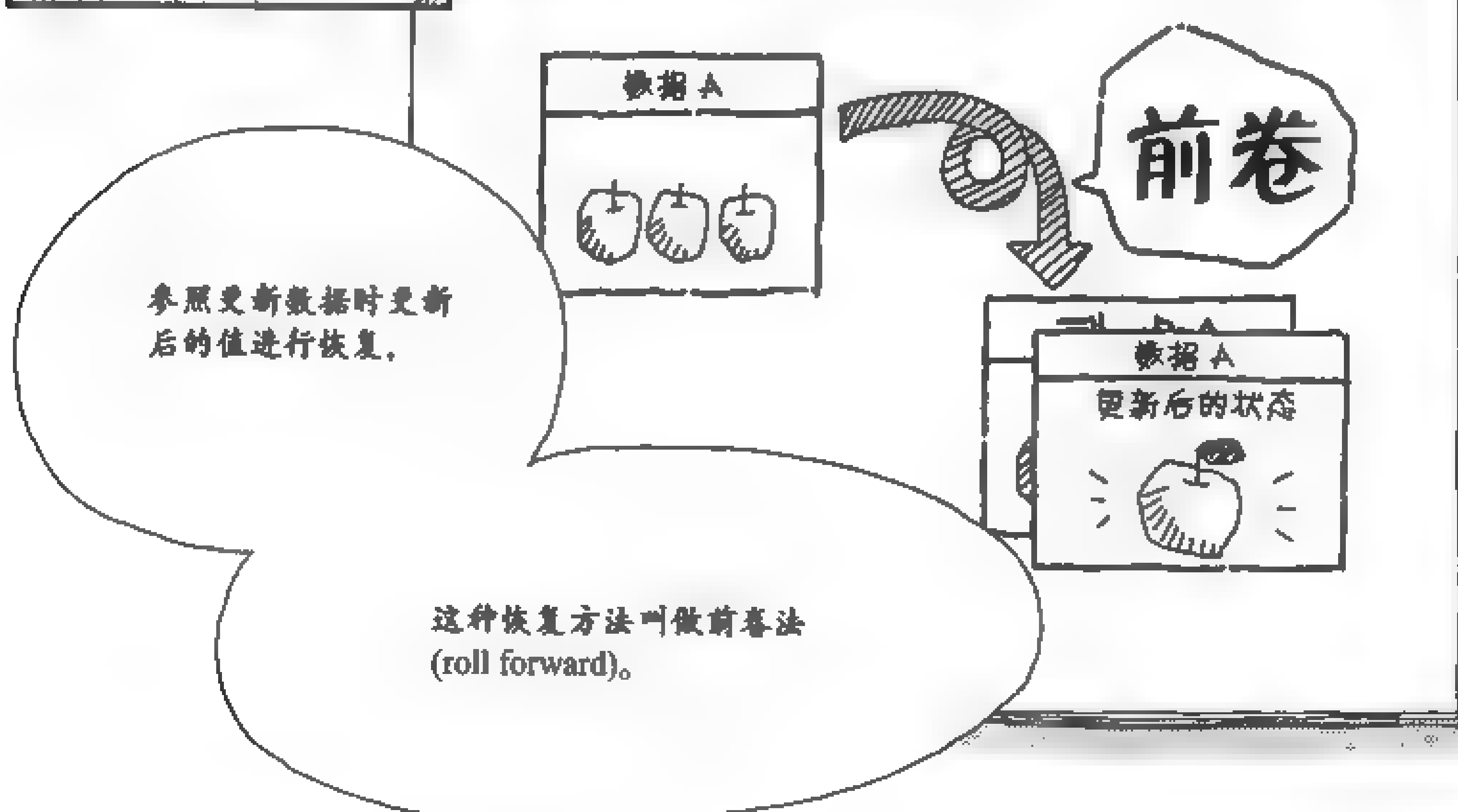
日志
数据库
日志
日志
日志
它记录了用户对数据所进行的变更等操作。

记录啊……
最重要的是在更新数据库时，记录了更新前和更新后的值。
嗯……
怎么样，小T？
悄悄地
……
嗯，那个人好像对数据库也产生兴趣了。
拉米内斯吗？
悄悄地
喂！

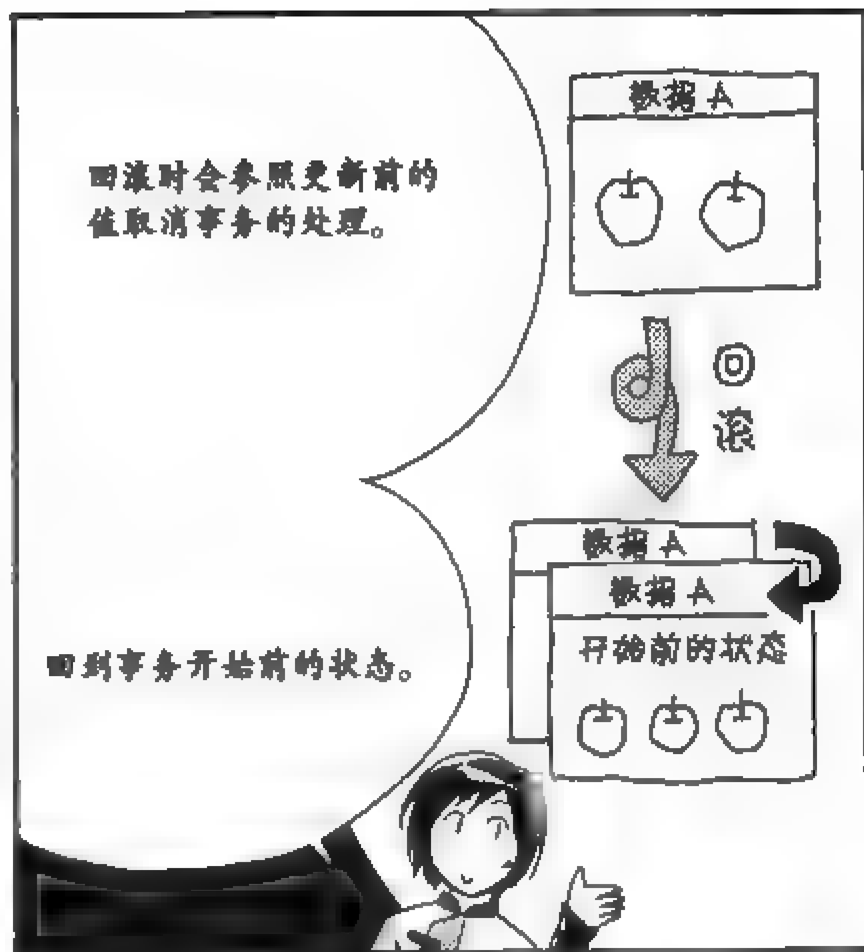
系统发生故障时首先要重新启动系统，
然后利用日志恢复数据库。
恢复的方法根据事务是否提交而不同。
好可怜呀……
事……提……
不继
好象不是很明白·
嗯

用图来解释的话……
不用了


首先，
故障发生时已提交的事务由于已经经过确认了，
让数据库反映这些处理进行恢复。


数据 A
前卷
数据 A
更新后的状态
参照更新数据时更新后的值进行恢复，
这种恢复方法叫做前卷法(roll forward)。

发生故障时没有提交怎么办呢？
对了对了
这时会进行回滚。


回滚时会参照更新前的值取消事务的处理。
回到事务开始前的状态。
数据A
回滚
数据A
数据A
开始前的状态

不仅正确地恢复了数据，而且数据还不会发生矛盾。
噢……

尽说些提交啊，事务啊之类的令人费解的词。
嗯……

好像数据库安全对策基本完成了呀！
ふぅ
不是吗？！

处理故障数据库能力很强啊！

那这次的功课就算了。
唉呀
可是……

结婚的事我是认真的！

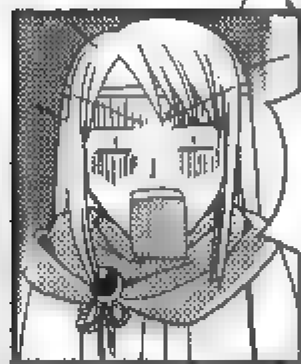
露娜…
拉米内斯，对不起！
我不能接受…

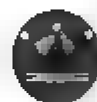
为什么呢？
因为我……
已经有喜欢的人了！

……是谁？
那个……

じっ
？

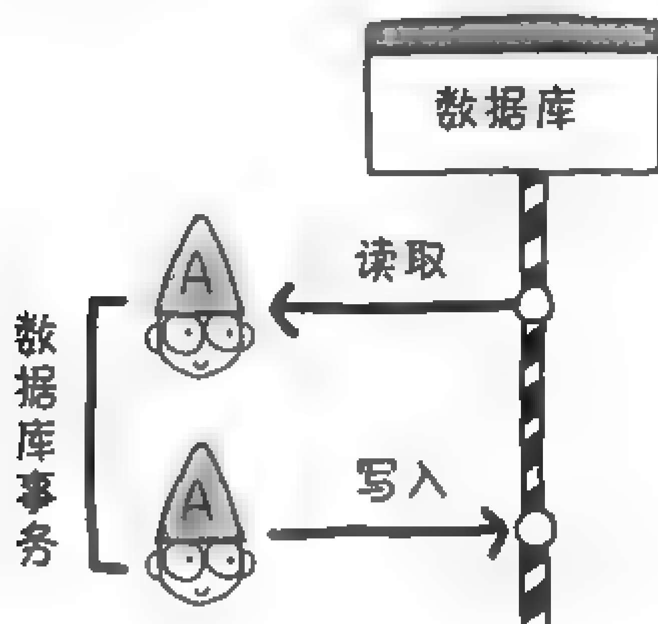
啊
ぎゅっ


我今后要和凯恩一起
使用数据库，
啊……啊？！
让编码王国更加繁荣！
对不起……
这，
这样啊——
是凯恩啊！！
是，嗯，
对不起。
为什么凯
恩要道歉
……
啊，啊，
对不起
……
以后请多关
照啊，凯恩！
是，是……
为什么呀？啊啊
两个人一起努力
的话，
一定会更好！



了解事务的性质

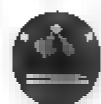
凯恩重复学习了数据库事务。使用数据库的用户可以利用数据库进行数据的检索、插入、更新、删除。用户的一个连贯操作我们称之为数据库事务 (Transaction)。



由于数据库的用户众多，即便多个数据库事务同时执行，也不允许各事务之间发生矛盾。另外，即使执行数据库事务的过程中发生了障碍，数据也不允许发生矛盾。因此，数据库事务应具有如下性质。要求数据库事务所具有的性质我们称之为 ACID 属性 (ACID PROPERTIES)。

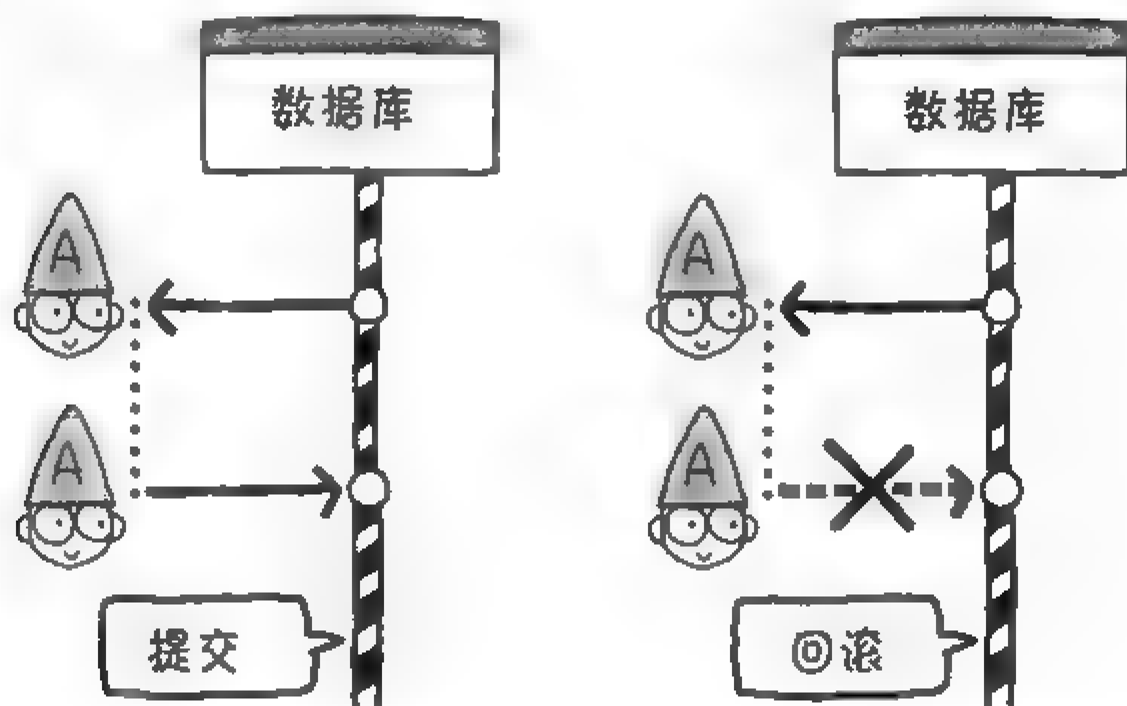
■ ACID属性

性　质	内　容	含　义
A (Atomicity)	原子性	数据库事务必须结束于提交或回滚中的任意一个任务
C (Consistency)	一致性	执行数据库事务时不能损坏数据库的一致性
I (Isolation)	隔离性	两个事务的执行是互不干扰的，一个事务不可能看到其他事务运行时，中间某一时刻的数据
D (Durability)	持久性	在事务完成之后，该事务对数据库所作的更改便持久的保存在数据库之中，并不会被回滚



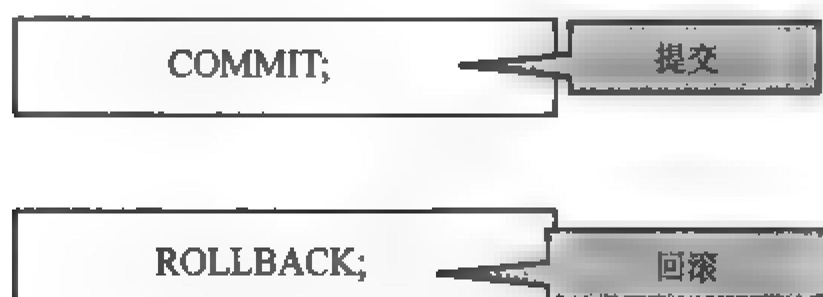
使用提交或回滚来结束

接下来让我们来看看这些属性。首先，数据库事务要具有原子性。数据库事务需由提交(commit)或回滚(rollback)来结束。提交是指确定数据库事务处理的指令。回滚是指取消数据库事务处理的指令。为了使数据库操作中不发生矛盾，数据库事务必须由提交或回滚中的任意一个指令来结束。



提交和回滚自动执行时，也有可能显式执行。显式执行的状况中，根据是否发生错误，变更数据库事务的处理。

在处理关系数据库的SQL中，有显式执行提交的COMMIT命令和显式执行回滚的ROLLBACK命令。



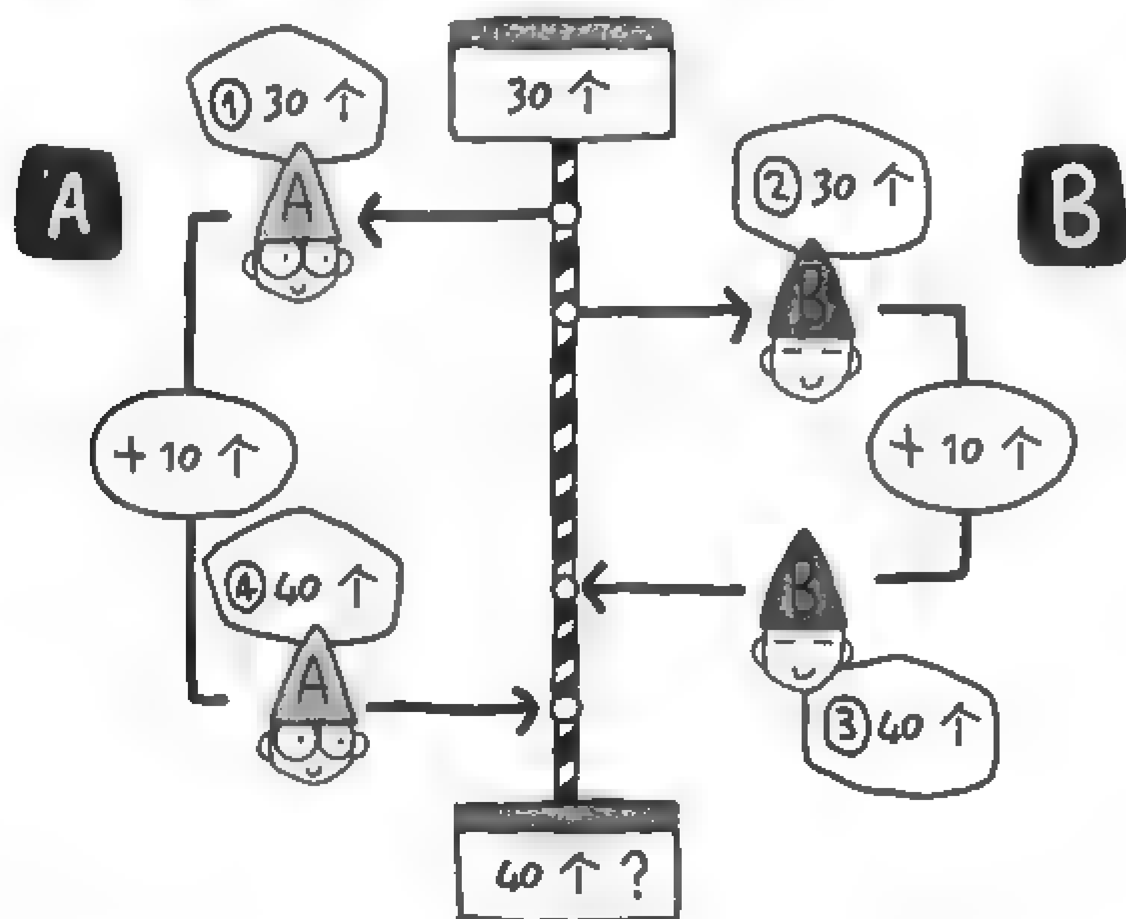
请写出确定数据库事务时使用的 SQL 命令。

请写出取消数据库事务时使用的 SQL 命令。

使数据不发生矛盾

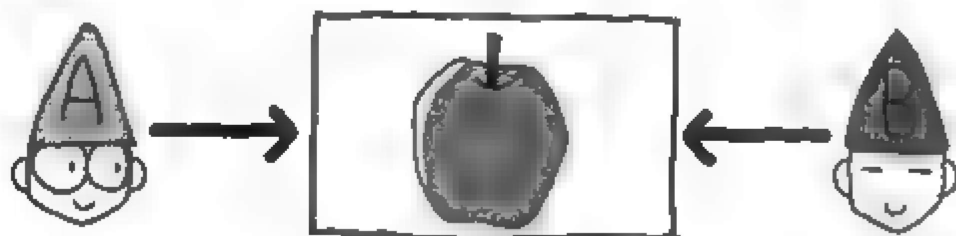
另外，数据库事务还需具有一致性。数据库事务执行前数据库不存在矛盾，操作执行后数据库也不能存在矛盾。

例如，凯恩提过的例子“数据库中有 30 个苹果，A 加 10 个，B 加 10 个，结果变成了 40 个的现象”。这种现象称之为“更新遗失 (lost update)”。



并行处理数据库事务时，多个事务可能同时访问相同的表格或行。此时，根据事务的处理顺序自然会发生矛盾。

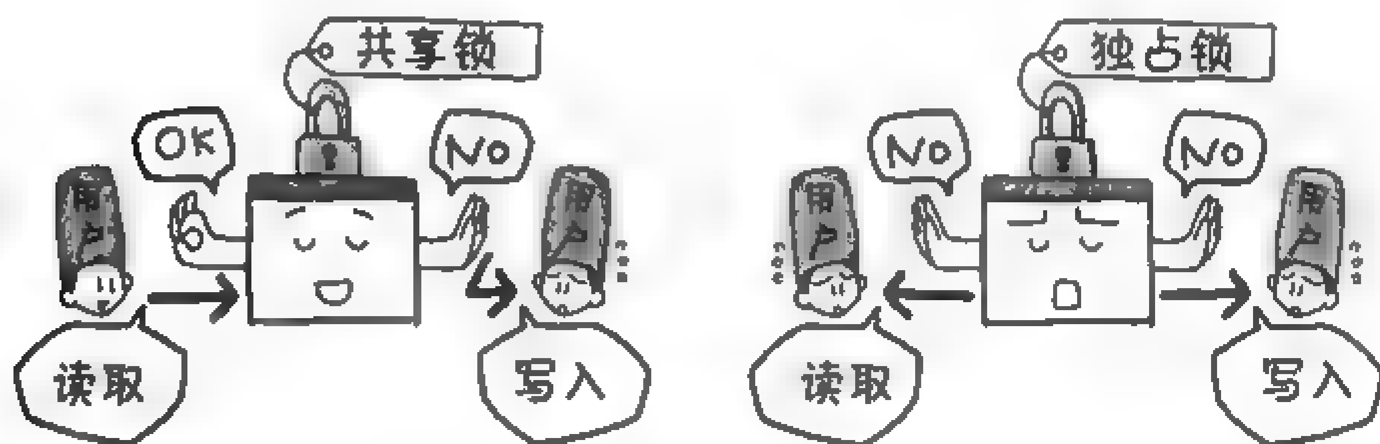
作为数据库事务操作对象的表格和行等单位我们称之为“资源”。在数据库中，事务即使并列访问相同的资源，也不会发生矛盾。



通过锁进行控制

即便是多个事务并行处理也和逐个处理得到的结果一样时，这个安排称之为“可序列化”(serializable)。数据库事务的隔离性要求安排必须可序列化。

由于安排可序列化，就有必要进行同时执行控制 (concurrency control)。这种方法最常使用的是由锁 (lock) 来进行控制。读取数据时使用共享锁 (shared lock)，写入数据时使用独占锁 (exclusive lock)。



使用共享锁时，其他事务可以加设共享锁，但是不能加设独占锁。使用独占锁时，无论是共享锁还是独占锁，其他事务都不可以加设。共享锁和独占锁的对立关系整理如下。

	共享锁	独占锁
共享锁	○	×
独占锁	×	×

A 加设了共享锁时，B 可以加设共享锁吗？

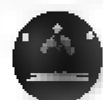
A 加设了独占锁时，B 可以加设共享锁吗？

A 加设了共享锁时，B 可以加设独占锁吗？

A 加设了独占锁时，B 可以加设独占锁吗？

B 加设独占锁失败。A 可能加设的锁型有哪些？

B 加设共享锁失败。A 可能加设的锁型有哪些？

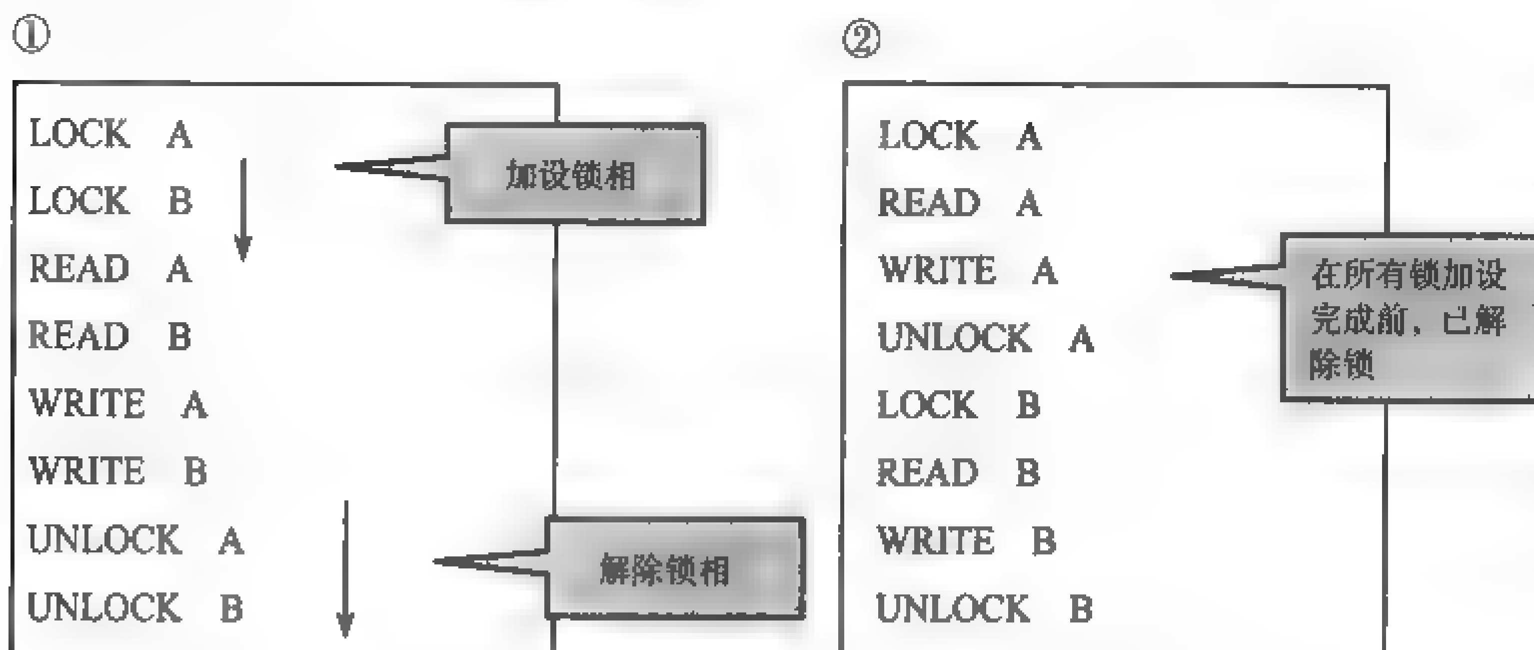


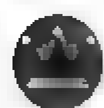
使用两相锁确保可序列化

在确保数据库事务可序列化过程中，对于锁的设定和解除需要遵守一定的规则。这个规则的其中一项就是两相锁。

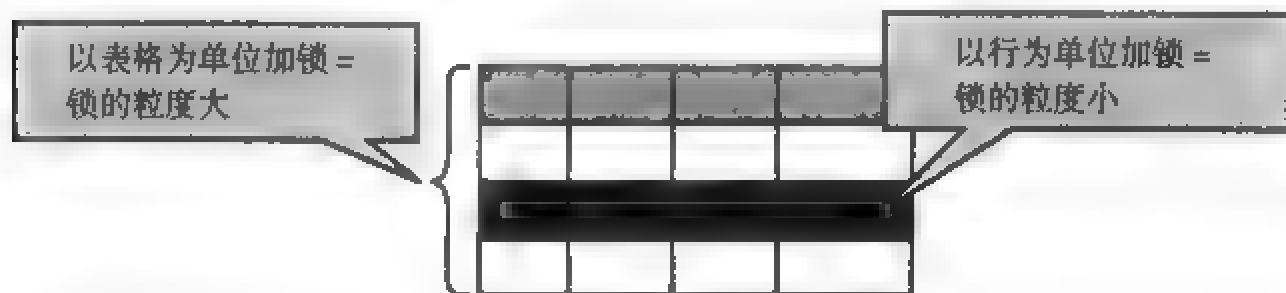
各事务是由加设锁和解除锁两相构成。

例如，有A和B这两个加设锁的对象。事务①使用了两相锁，但事务②没有使用两相锁。各个事务根据遵守两相锁的情况保证了他们的可序列化。



 注意锁的粒度

作为加设锁的对象要考虑到多种类型。例如，以表格为单位加设锁时，要考虑到以行为单位加锁的情况。加锁的范围我们称之为锁的粒度 (granularity)。



增大锁的粒度，每个事务加设锁的次数降低，这样可以减少管理锁的工作量。因此，CPU在运转数据库时的负担就会小一些。但是，由于加锁的对象很广泛，等待解除那些其他加锁事务的时间就会很长。因此，同时可以执行的事务就减少了。

相反，减少锁的粒度，每个事务加设锁的次数增加，从而增加了管理锁的工作量。因此，CPU的负荷也就增加了。但是由于加锁的对象范围很窄，等待解除其他加锁事务的时间也就减少了。因此，可同时执行的事务数也就增加了。

将加锁的对象从表格变更为行，那么可同时执行的事务数如何变化?

将加锁的对象从行变更为表格，那么可同时执行的事务数如何变化?

其他同时执行控制

通过锁来控制操作进程，是为了同时执行多个事务。但是，使用锁来进行同时执行控制，存在管理锁的负担。另外，还有可能出现死锁的现象。

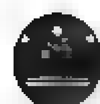
那么，在事务数很少的情况下或读取行为多的情况下，可以使用更为简洁的方法进行同时执行控制。这种方法有以下几种控制方式。

时间戳控制 (Timestamp control)

数据库事务给每一个被访问的数据打上了一个“时间戳”的时间印记。时间戳控制法是指某个事务要读写这个数据时，比这个事务拥有更早时间戳的事务更新了数据的情况下，不允许读写数据的方法。不许读写的情况下，回滚此事务。

乐观控制 (Optimistic control)

是一种暂时允许各事务读取的处理方法。从写入点开始，确认是否由其他的事务更新了数据。若其他事务更新了数据，则回滚。



隔离级别的设置

在现实的数据库中，同时执行的事务不断增加，因此能够逐渐控制事务之间互相干涉的级别。这叫做隔离级别 (isolation level)。

使用 SQL 中的 SET TRANSACTION 命令即能够设定以下事务的隔离级别。

- READ UNCOMMITTED
- READ COMMITTED
- REPEATABLE READ
- SERIALIZABLE

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
```

通过设定隔离级别，可能发生以下现象。

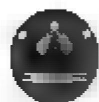
	脏　读	非重复读	虚　读
READ UNCOMMI TTED	可能发生	可能发生	可能发生
READ COMMITTED	不发生	可能发生	可能发生
REPEATABLE READ	不发生	不发生	可能发生
SERIALIZABLE	不发生	不发生	不发生

脏读 (dirty read) 是指事务 1 在提交前事务 2 读取该行，在事务 1 回滚的情况下，事务 2 读取了不存在的行这种现象。

非重复读 (non-repeatable read) 是指事务 1 读取行时，事务 2 在更新该行并提交时，事务 1 再一次读取该行时发生数值不一致的现象。

虚读 (phantom) 是指事务 1 进行检索，获得多行结果，事务 2 追加了符合该条件的行，事务 1 第二次检索的结果发生不同的现象。

未设定隔离级别的情况下，默认为 SERIALIZABLE。



数据库的安全问题

数据库管理着重要的数据。因此确保数据库的安全就显得非常重要。确保安全对于防止用户随意更改数据而导致矛盾的产生也很重要。

在使用数据库时，能够设定访问数据库和表格的权限。凯恩通过强化数据库安全，避免了王国的数据库危机。

在关系数据库中，SQL 命令中有给予用户操作权限的 GRANT 命令。通过 GRANT 命令，可以给予其他用户操作管理者和用户生成的表格的权限。设定权限是运用数据库过程中非常重要的工作。

```
GRANT SELECT, UPDATE ON 商品 TO 外国部；
```

赋予操作权限

使用 SQL 命令的话，可以授予以下权限。

■ 权限示例

SELECT	检索表格中行的权限
INSERT	向表格中插入行的权限
UPDATE	更新表格中行的权限
DELETE	删除表格中行的权限
ALL	所有权限

附带 WITH GRANT OPTION 授予权限时，被授予权限的用户可以再授予其他用户权限。按照如下命令授予权限时，外国部可以授予其他用户检索、更新的权限。

```
GRANT SELECT, UPDATE ON 商品 TO 外国部
WITH GRANT OPTION；
```

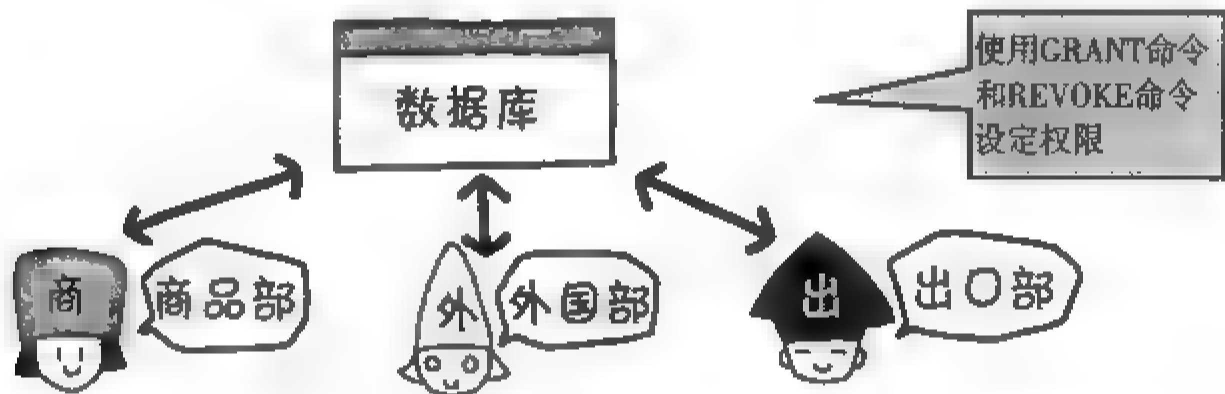
能够授予其他用户权限

另外，也可以收回 GRANT 命令授予的权限，此时使用 REVOKE 命令。

```
REVOKE SELECT, UPDATE ON 商品 TO 外国部；
```

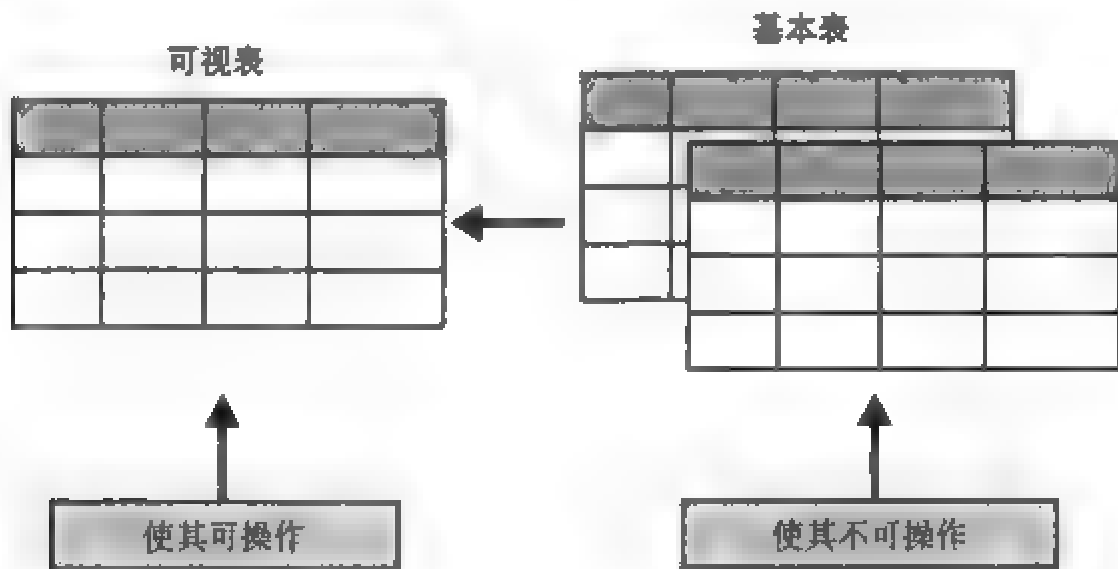
能够收回权限

根据数据库产品的不同，也可以将多个权限归纳形成组，并集中授权。权限组群化使得权限的管理更加容易。



另外，使用第 4 章第 119 页中介绍的可视表，能够在安全性上进行更加细化的管理。

首先，抽取基本表的一部分制作可视表。通过设定对该可视表的权限，能够实现设定访问表格中部分数据的权限。进行安全设定对于保护数据库中的数据是非常重要的工作。



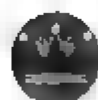
请写下授予出口部检索商品表数据权限的 SQL 命令。

请写下收回外国部删除商品表数据权限的 SQL 命令。

对管理员制作的商品表设定了以下权限。在下表中打○和 ×，标注出每个部门有无该权限。

```
GRANT ALL ON 商品 TO 外国部
GRANT SELECT, UPDATE ON 商品 TO 商品部;
GRANT SELECT, INSERT ON 商品 TO 出口部;
```

	检索	插入	更新	删除
外国部				
商品部				
出口部				



使用索引进行快速检索

数据库管理着大量的数据。在从大量的数据中检索部分数据时，需要一些处理时间。

商品编码	商品名称	单价	地域
101	香瓜	800G	南部
102	草莓	150G	中部
103	苹果	120G	北部
104	柠檬	200G	南部
201	栗子	100G	北部
202	柿子	160G	中部
301	桃子	140G	南部
302	猕猴桃	200G	南部

全部检索的话非常耗时

因此，为了使数据库的检索能快速进行，可以设定索引 (index)。索引是快速访问数据收纳位置的手段。在从大量的数据中选择其中一部分数据的情况下，使用索引可以进行快速检索。

商品编码	商品名称	单价	地域
101	香瓜	800G	南部
102	草莓	150G	中部
103	苹果	120G	北部
104	柠檬	200G	南部
201	栗子	100G	北部
202	柿子	160G	中部
301	桃子	140G	南部
302	猕猴桃	200G	南部

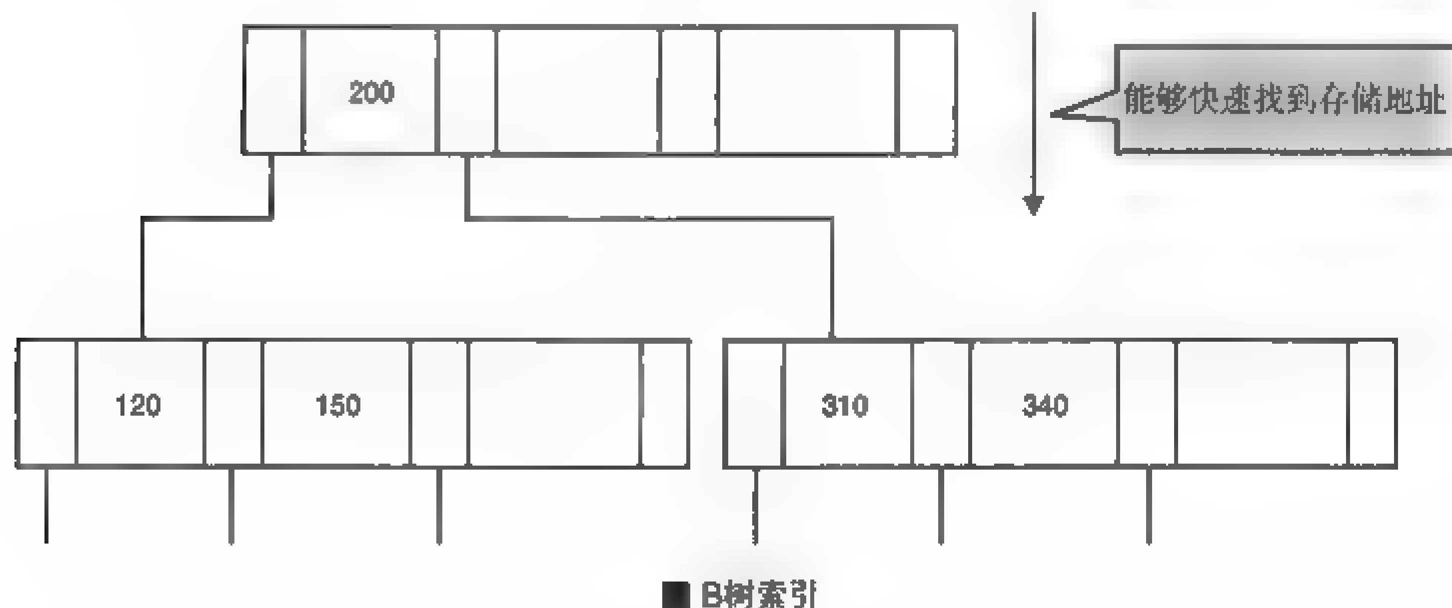
使用索引可快速访问存放地址

索引部分

索引方法有 B 树、散列等。

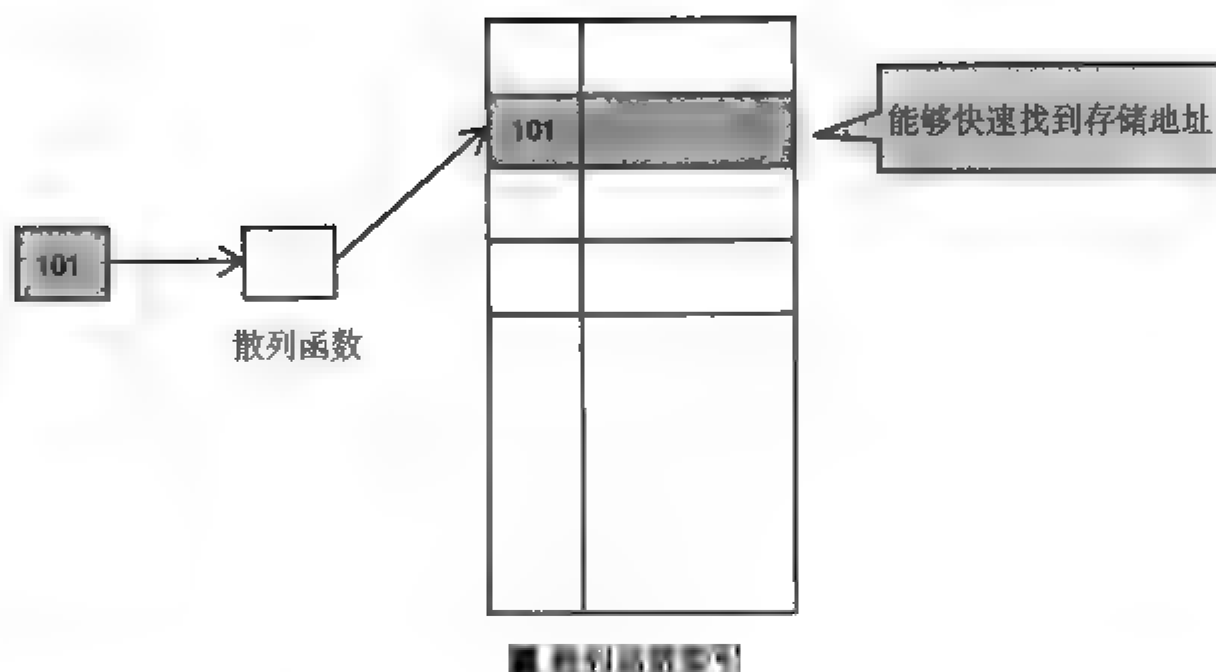
B 树 (B tree) 是一种以树的构造管理存储位置信息的方式。在 B 树中每个节点 (node) 可以有多个子数据，从而控制树的高度。因此，即使有很多数据，也可以很快地从根部找到存放地址。

B 树中除根节点外，其他节点拥有的子数据最多为 2n 个，最少为 n 个。



■ B树索引

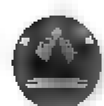
散列 (hash) 是对数据键值运用散列函数，求得存储地址的方法。散列用于“商品编码为 101”这样的完全一致检索时功能强劲。但是，散列不适用“商品编码为 101 以上”这样的比较条件检索和“末尾带‘子’的商品”这样的模糊检索。



即使使用索引，有时也不一定能快速处理。例如，在抽取所有数据时，使用索引会造成处理延迟。另外，更新数据时，需要重新制作索引，会造成更新速度缓慢。

在使用等号检索过程中，B 树和散列哪个功能更强？

在使用不等号检索过程中，B 树和散列哪个功能更强？



最优化查询

查询数据库时，要在解析 SQL 查询内容的基础上，考虑索引等方式以达到快速查询的目的。因此，让我们来看看查询的实质。

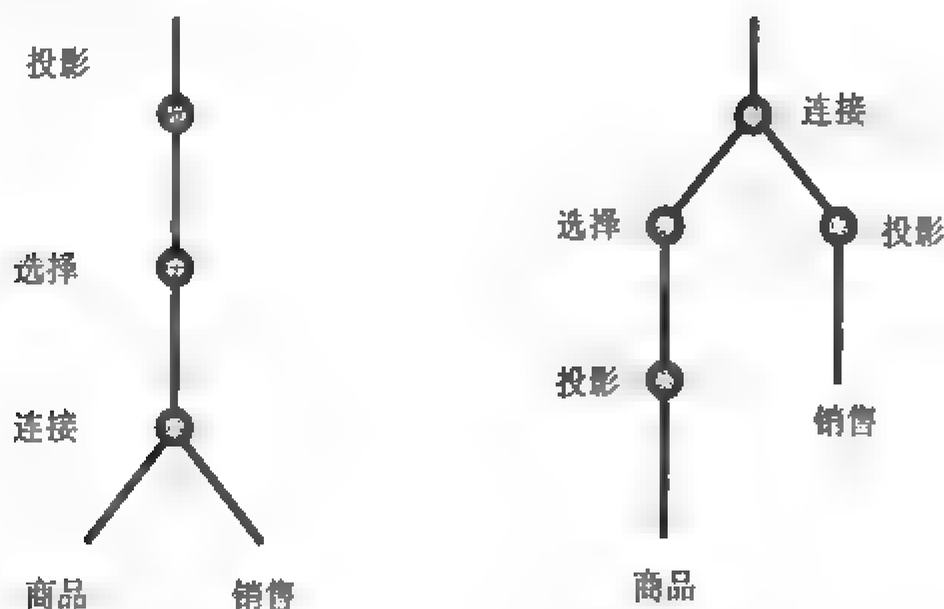
首先，在进行查询时，要解析查询内容，按照适当的顺序操作。例如，我们进行“抽取单价在 200G 以上的商品销售日期、商品名称”这样的查询，此查询由以下三步构成。

```
SELECT 日期，商品名称
FROM   商品，销售
WHERE  单价 >=200
AND    商品.商品编码 = 销售.商品编码
```

① 连接商品表和销售表

② 选择单价在 200G 以上的商品

③ 抽取日期和商品名称列



■ 最优化查询

例如在上图左侧，从下往上看是按照 1 → 2 → 3 的顺序进行查询的，与此相对，右侧是按照 3 → 2 → 1 的顺序进行查询的。按照任何一个顺序都可以进行相同的查询。

但是，按照 1 → 2 → 3 的顺序，在最初的连接过程中生成了行数很多的中间列表，一般处理时间比较长。相反，按照 3 → 2 → 1 的顺序，由于先行进行选择和投影，消除了行和列，所以一般处理时间比较短。

这样，相同的查询，根据投影、选择、连接等的不同，运行顺序也不同，处理的时间也不同。

一般地按照：

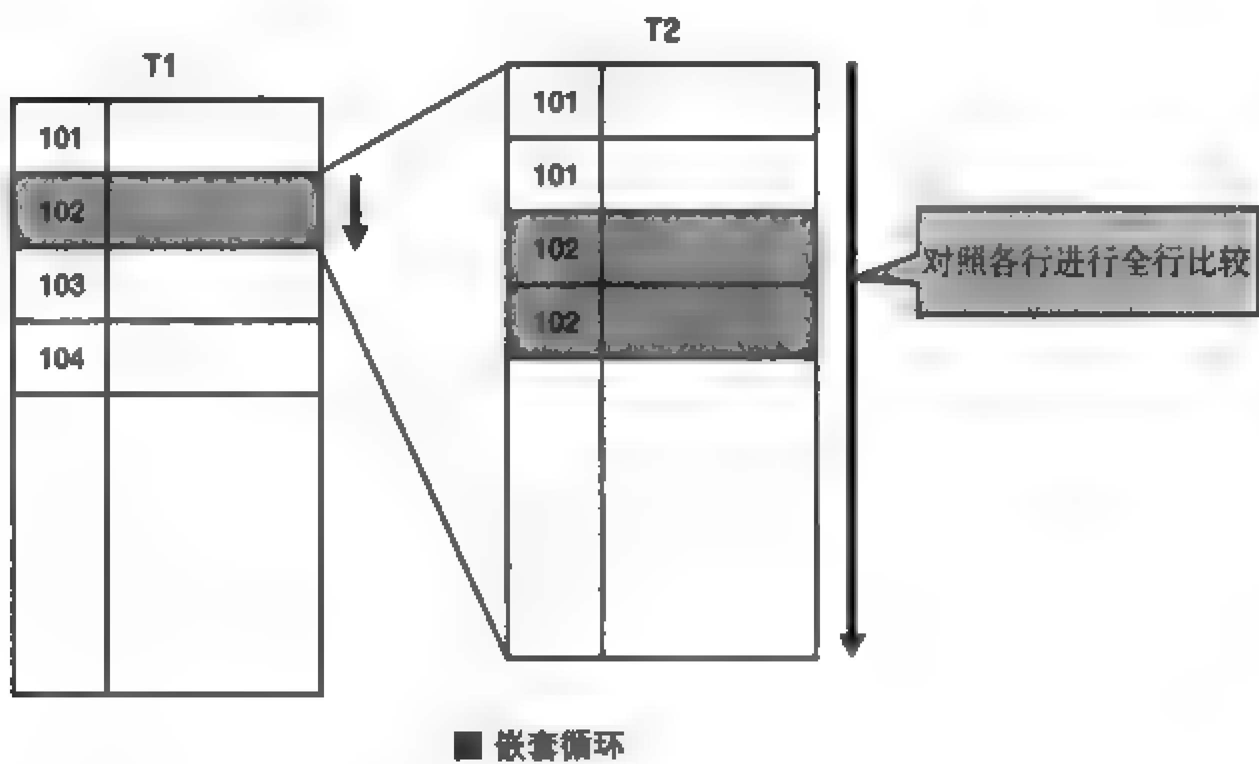
- 先行执行选择减少行数
- 再执行投影减少与结果无关的列
- 最后执行连接

这样的标准确定查询顺序。

在执行投影、选择、连接时，存在多种技巧。例如，选择方法有全条件检索方法和索引检索方法。另外，连接方法有如下几种。

嵌套循环 (nested loop)

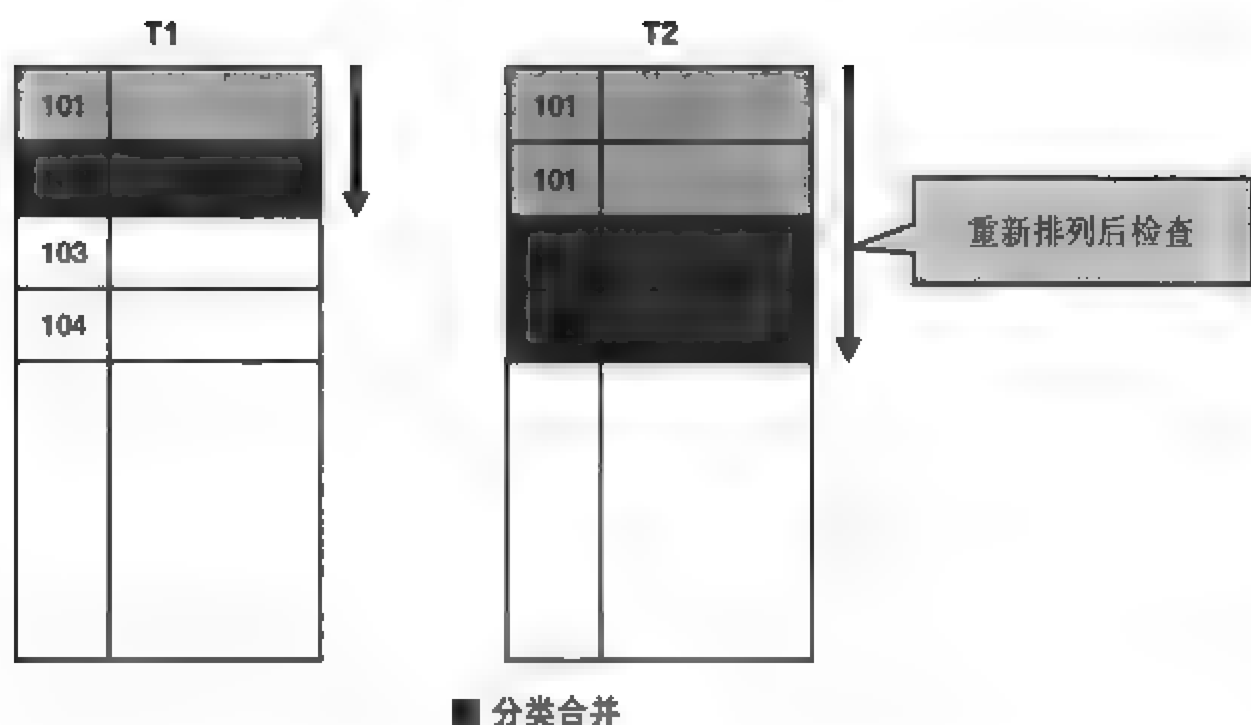
一张表格的一行比较另一张表格所有行的方法。(参照下图)例如,从表T1中抽取 -行，表T2中列值与之比较是否相同。相同的情况下，将作为结果生成连接后的行。



■ 嵌套循环

分类合并 (sort merge)

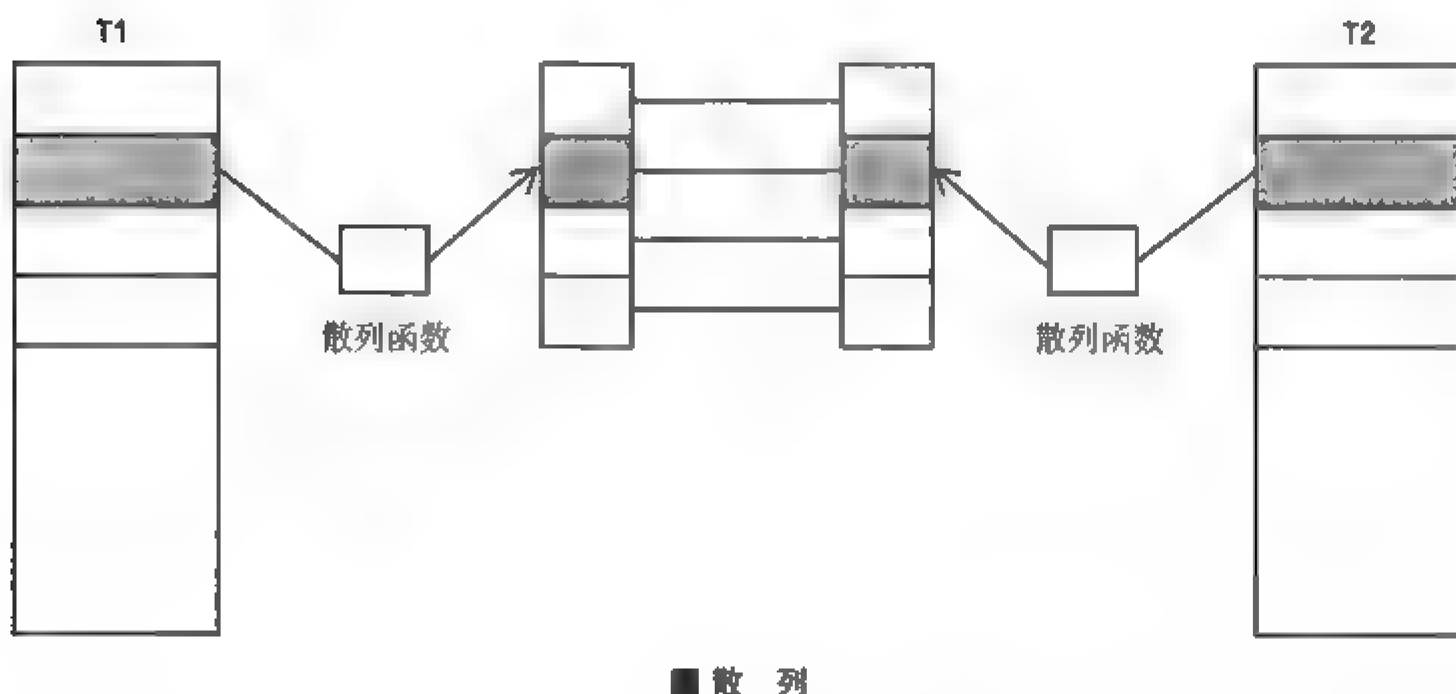
是指重新排列各表格的行后，进行连接的方法(参照下页中的图)。首先将表T1和表T2全部或部分进行分类。之后，从第一行开始检查，值相同时作为结果生成连接后的行。由于要进行分类，进行一个方向的处理就够了，所以处理时间非常短。但是，需要注意先行分类所花费的时间。



■ 分类合并

散　列 (hash)

是指将一张表格根据散列函数进行分割后，与另一张表格中具有相同散列值的行进行连接的方法。这种方法求取连接行的效率很高。



■ 散　列

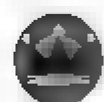
查询时应最优化使用这些方法。数据库中使查询最优化的功能叫做优化程序(optimizer)。最优化标准有以下几种。

规则导向 (rule based)

预先确定多个规则，按照规则决定的优先顺序选择方法的方式。例如，在确定了使用索引访问数据的情况下，即使进行整体检索，也应使用索引进行检索。因此，有可能无法根据数据库的状态进行检索。

成本导向 (cost based)

根据数据库内部的统计信息，选择方法的方式。即定期生成了数据分布等统计信息，再根据这些信息进行选择的方式。虽然相比规则导向方式，成本导向方式更可能在灵活性方面达到最优化，但是需要定期生成统计信息。因此，也需要在管理和分析统计信息上花费工夫。

 故障恢复

数据库拥有即使发生故障，数据也不会出现矛盾的结构，这是非常必要的。事务的持久性 (durability) 要求不能出现因为故障发生错误结果。为了防止发生故障，数据库需可执行获取备份文件和事务日志的功能。

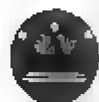
数据库的故障是由各种状况引起的。不同的状况可能产生以下故障：

- 事务故障
- 系统故障
- 介质故障

事务故障是指因为事务不完备而导致事务不能结束的情况。在事务故障中，发生故障的事务将被回滚。

系统故障是指因为停电等原因造成系统停运的情况。在系统故障中，重新启动后进行故障恢复处理。通常，对故障发生时未提交的事务进行回滚，对故障发生时已提交的事务进行前卷。

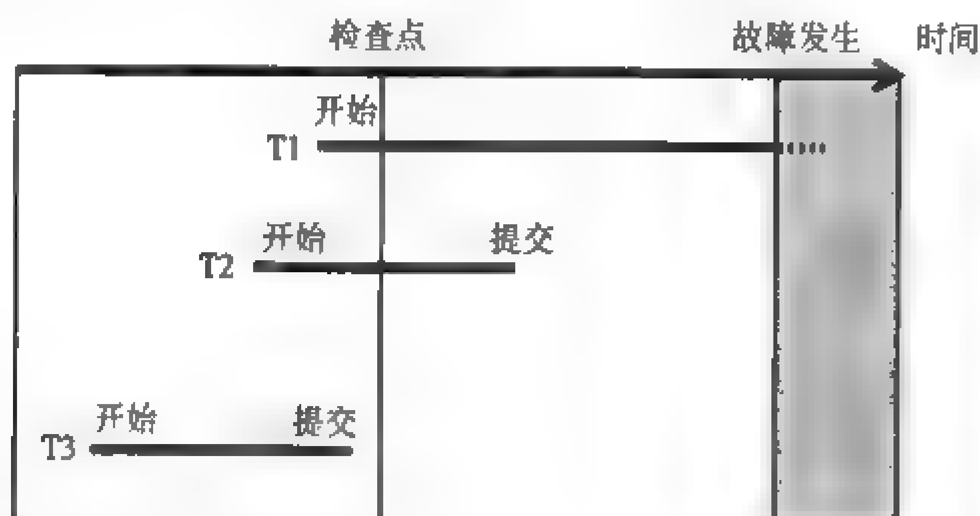
介质故障是指硬盘损伤的情况。发生介质故障时，可以基于备份文件进行故障恢复。对备份后提交的事务进行前卷处理。



检查点和恢复

另外，在实际应用的数据库中，为了提高向数据库中写入的效率，多采用暂时将数据写入缓存的方法。写进缓存的内容与检查点 (checkpoint) 中的数据内容是一致的。采用这种方法时，检查时对已提交的事务是不必进行故障恢复的。检查时未提交的事务才是故障恢复的对象。

另外，在系统故障中执行了如下事务。哪个事务应回滚、哪个事务应前卷呢?



T1 如何处理?

T2 如何处理?

T3 如何处理?

即使数据库发生故障，使用这里介绍的恢复处理功能，构成了可使数据库不发生矛盾的构造。这样自然就可以安全地使用数据库了。

小结

- 能够设定数据库用户的权限。
- 同时执行控制方法——锁。
- 通过生成索引能够进行高速检索。
- 数据库具备故障恢复功能。

答案

A1 COMMIT；

A2 ROLLBACK；

A3 ○

A4 ×

A5 ×

A6 ×

A7 共享锁或独占锁

A8 独占锁

A9 增加

A10 减少

A11 GRANT SELECT ON 商品 TO 出口部；

A12 REVOKE DELETE ON 商品 FROM 外国部；

A13

	检索	插入	更新	删除
外国部	○	○	○	○
商品部	○	×	○	×
出口部	○	○	×	×

A14 散列

A15 B树

A16 由于故障发生时尚未提交，所以进行回滚。

A17 由于故障发生时已提交，所以进行前卷。

A18 由于检查时已提交，所以没有必要进行恢复处理。

第6章

数据库的普及和灵活应用

哈哈哈!!

还是编码王国的水果好吃！！

父亲！！

嗯？怎么了？露娜也要吃？

不是的！

爸爸回来后就只知道吃水果……

呵呵，不好吗？

外出的这段时间，多亏了露娜精心照料，编码王国才井井有条！

的确是这样的！！数据库真方便啊。

哈哈哈

呵呵呵

除此之外还有别的什么用途吗？
嗯……嗯，那个

小……
小T不见了？
小T？

咦？……
没见过……
那个，小T是一个这么高，能在空中飞的女孩……
是从爸爸拿回来的那本书里出来的！！

爸爸妈妈回来前还在呀！
哈哈

见过吗？
没有
是爸爸拿回来的书，我还以为他知道呢。
原来除了我和凯思谁都看不到啊……
嗯——

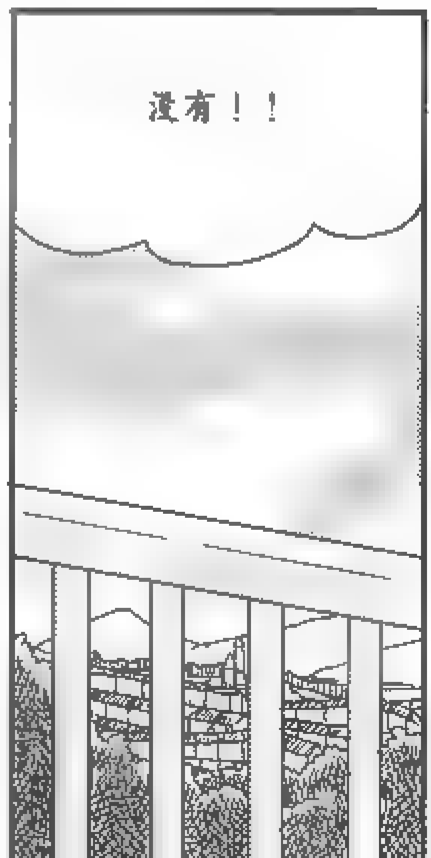
露娜公主！！

确实不见了！
打开书也不出来。

去哪儿了呢？
我们再去找找看吧！

对不起
啊，失礼了。
嘿
嘿

咦，这俩人怎么了？
好像关系很不一般嘛？
凯恩快点！！
好，来了！！
嗯……
呵
呵


没有！！

哪儿都没有……

小T好像没有突然失踪过…

…

连告别都……

不辞而别了

是呀…

在找谁呢？

！！！

?!

这里，这里……

啊！

小T！

哈哈！

不要得意啦！
露娜好可怕哟
哼——!!
跑到哪里去了？
没什么，有点忙而已····

好担心你啊……

····
对不起！

现在编码王国差不多都用上数据库了！
嗯！

托你的福，效率显著提高了！

大海对面有国家用数据库管理更多的事情呢。
哦！

数据库的应用案例

例如……
有的国家用在银行的储蓄账户管理系统中。

欢迎光临

银行和数据库吗？

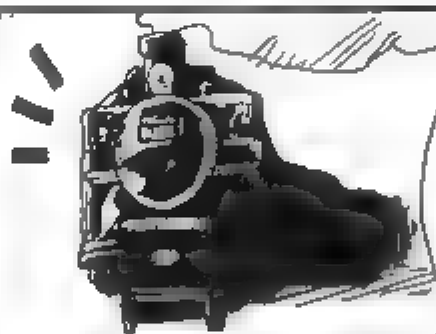
储蓄系统可以让很多人使用的话……

取款

转账

可以自己取款或给别人转账了呀！

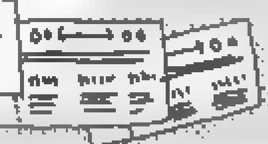
那真是很方便呀！



也有将数据库用于火车票预订系统的。

嗨

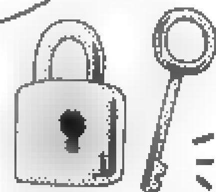
预约卡



使用了数据库无论去哪站都可以预约座位了。

是啊，是啊！

之前我们也学习了使用锁，对吧？



lock

是啊，
使用锁可以控制事务，对吧？
是的，是的！
A和B这两个家伙

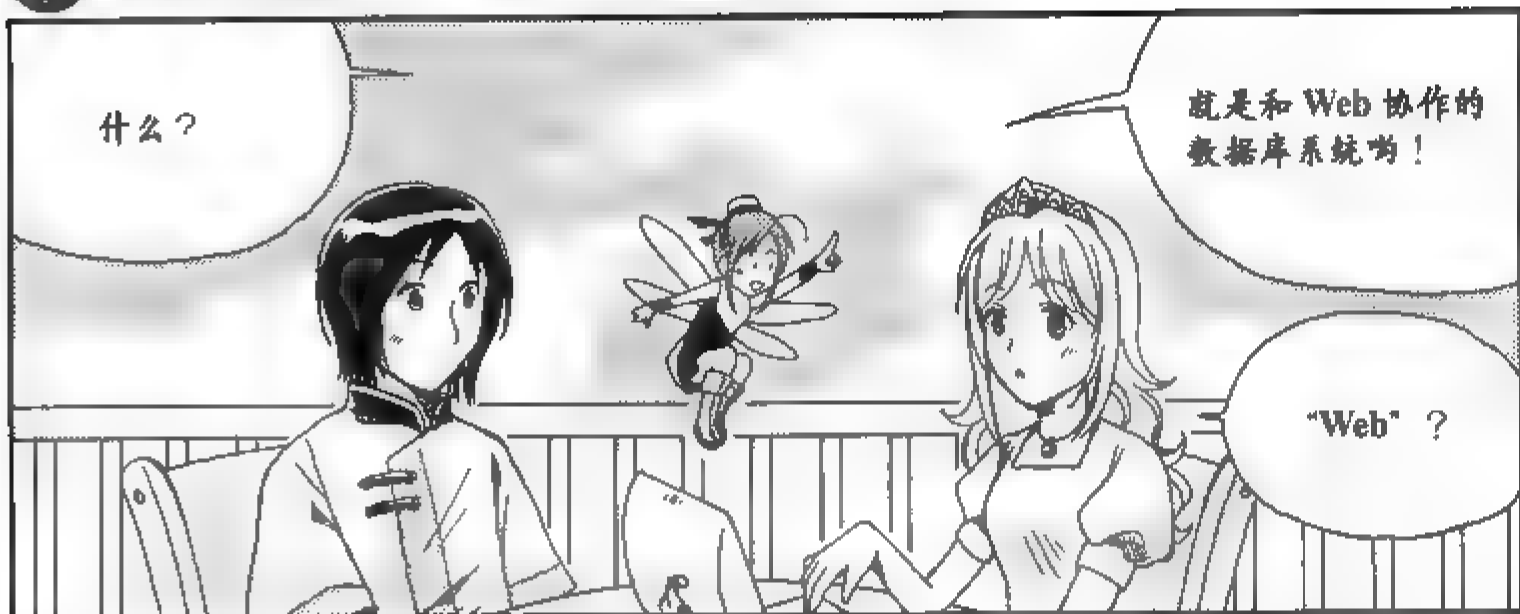
由于数据库拥有这些功能，所以即使很多人从其他站同时访问，
预订的座位也不会重复。
预订
数据库

这么说也考虑到安全和故障对策了。
安全
数据库
NO!!
OK!!
故障对策
是啊，是啊！
在储蓄数据库中是绝对必要的功能。
谁都可以随便取款的话就麻烦了，不是吗？

数据库可以用于各种事务。
存款不见了就麻烦了
就是呀！

最近的数据库有了更广泛的用途。

Web与数据库


就是和 Web 协作的数据库系统哟！
什么？
"Web"？

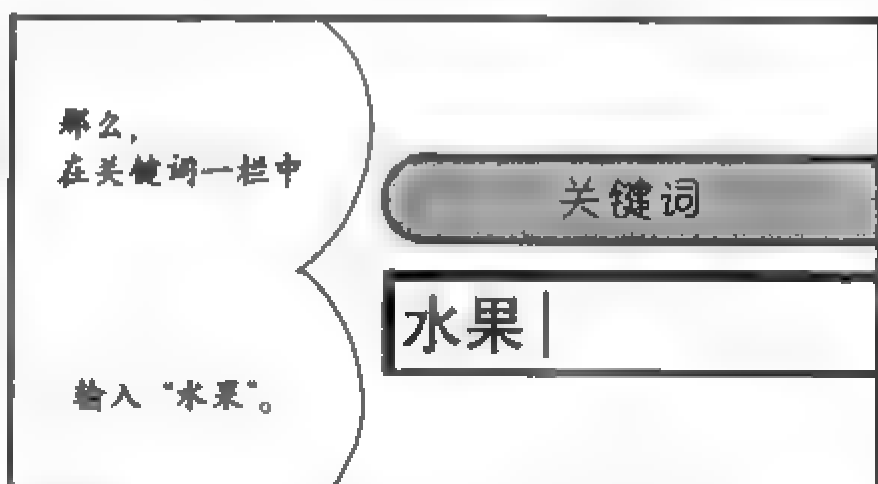
在大海的那边儿可以一边上网一边买各种东西了。
啊，想要这本书
买入
好，买了！
收到了！

比如上网浏览，
就可以买自己想要的书了。
那真是方便呀！！真好啊！
真棒啊！！

不用根据书单一一书店地去买了！！
真想也用在这方面啊！！
国王的需求
嗯 下一个

要找想要的书时，
砰
Book search
小丁查询
所有商品
关键词
在Web浏览器中
输入关键词，
小丁查询

要找什么书呢？
嗯，
仍旧是"水果"吧……


那么，
在关键词一栏中
输入"水果"。
关键词
水果

这个关键词将作为HTTP
请求被送出，
接受请求并进行处理的电
脑叫做服务器(sever)，
Book search
所有商品
关键词
水果
在服务器上生成SQL。

服务器

SQL命令

```
SELECT 商品名
FROM   商品
WHERE  商品名
       LIKE '%水果%';
```

数据库服务器

数据库

路由器

互联网

路由器

检索条件

水果

商品数据

わ～

然后将此SQL命令发送至数据库进行查询。

Web页面
服务器
数据
数据库服务器
数据库
读取
路由器
互联网
路由器
检索结果
哇，出来了！！
原来如此！

这样就可以在 Web 页面上看到与水果相关的书的列表了。

如果想买商品，进行相同的处理就可以了！

这时，只要使用 SQL 命令从“商品库存”表中减少库存数量，
在“商品配送”表中添加数据就行了。
喷？
哦……
仍旧使用的是 SQL 呀。
露娜公主……

看……这是什么？
搜索
哦？啊！！

什么嘛，这是！
露娜的父亲什么时候……
还能找到这么意外的东西啊？

可是 Web 上的书店也会有很多顾客同时光顾，不是吗？

Web书店

即使使用锁和安全等功能来确保数据库安全，

处理也很麻烦，是吧……

这个时候会有很多台服务器来共同处理哟！

使用好几台服务器吗？

就是 Web 服务器 (Web server) 和应用服务器 (application server) 了。

Web服务器

应用服务器

"Web 服务器"就是那个生成 Web 页面的服务器吧……

Web服务器

Web页面

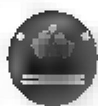
"应用服务器"是进行 SQL 命令组合作业的服务器。

应用服务器

SELECT

FROM

WHERE



分布式数据库

数据库服务器也可以分开吗？

那叫分布式数据库(distributed database)。

由多个服务器来管理数据库，是吗？

是的！

数据库

服务器

服务器

服务器

但是必须把它们当做一个整体的数据库来处理。

如果能把多个服务器管理的数据库当成一个来管理那可真是厉害呀！

按照服务器的功能来管理就行呦！

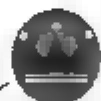
能够有效抵御故障！

故障

Knock Out!!

数据库

即使部分数据库发生故障，整体数据库也不会瘫痪。



存储程序和触发器

在任何环境中都使用多个服务器的情况下，必须好好利用网络。
是的！为了减少带给网络的负担，
需要登录一个叫做存储程序(stored program)的程序。
存储？
啊？
存储（store）不就是记录下来的意思吗？
是的！！
就是将为了减少带给网络的负担而经常使用的处理存储在数据库中。
经常使用的处理…
都是些什么处理呢？
嗯……
嗯，例如买书时的处理
"从库存表中减少库存，在配送表中添加数据……"
这些处理怎么运作呢？

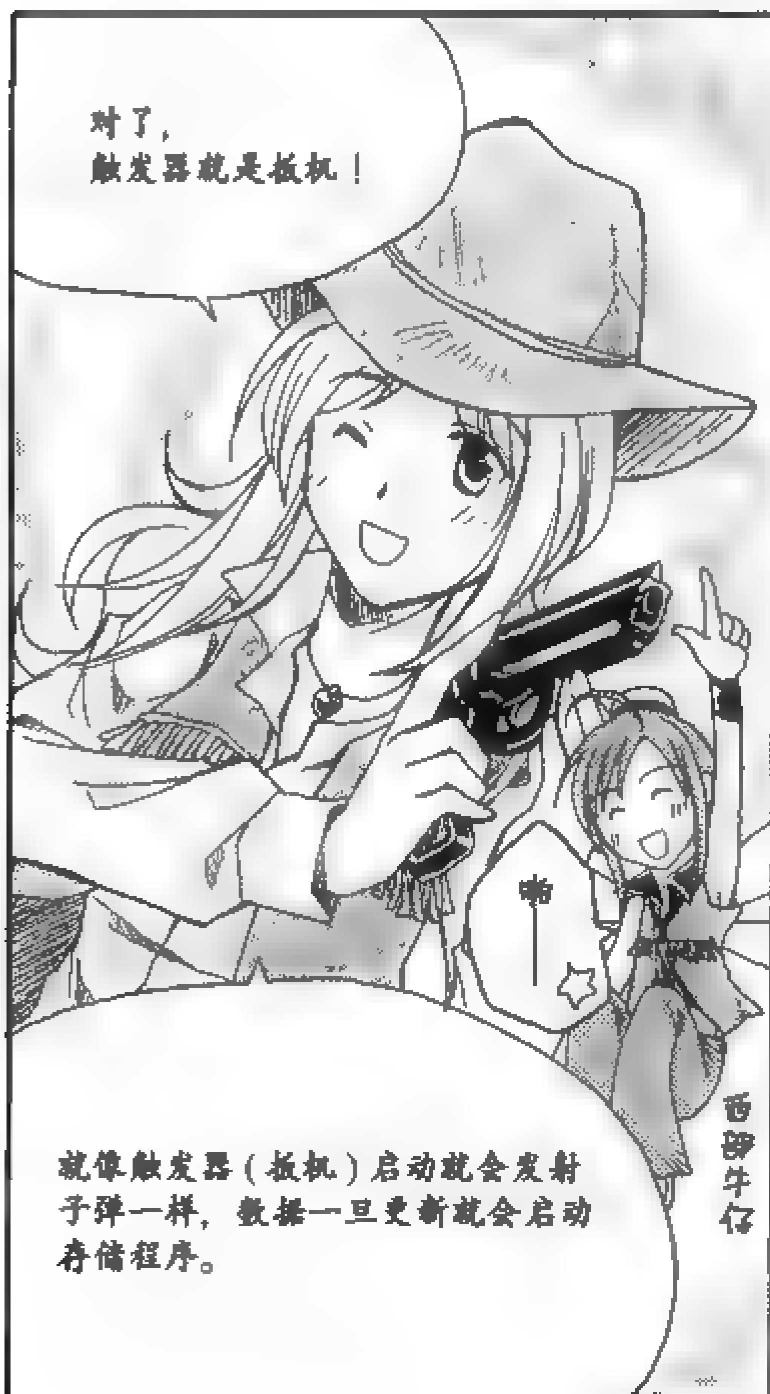
就是将所谓的每次都使用的处理当做预存步骤，
嗯，是啊！
登录在数据库中。

生成存储程序后，"减少库存进行配送" 这些处理就不用一个一个地发送 SQL 命令了，
一步就可以完成了。
原来如此
网上的处理也就变得轻松了。

而且我们的工作也越来越轻松了呀。
啊！！
对啊！！

还有自动启动的存储程序呢。
自动的？

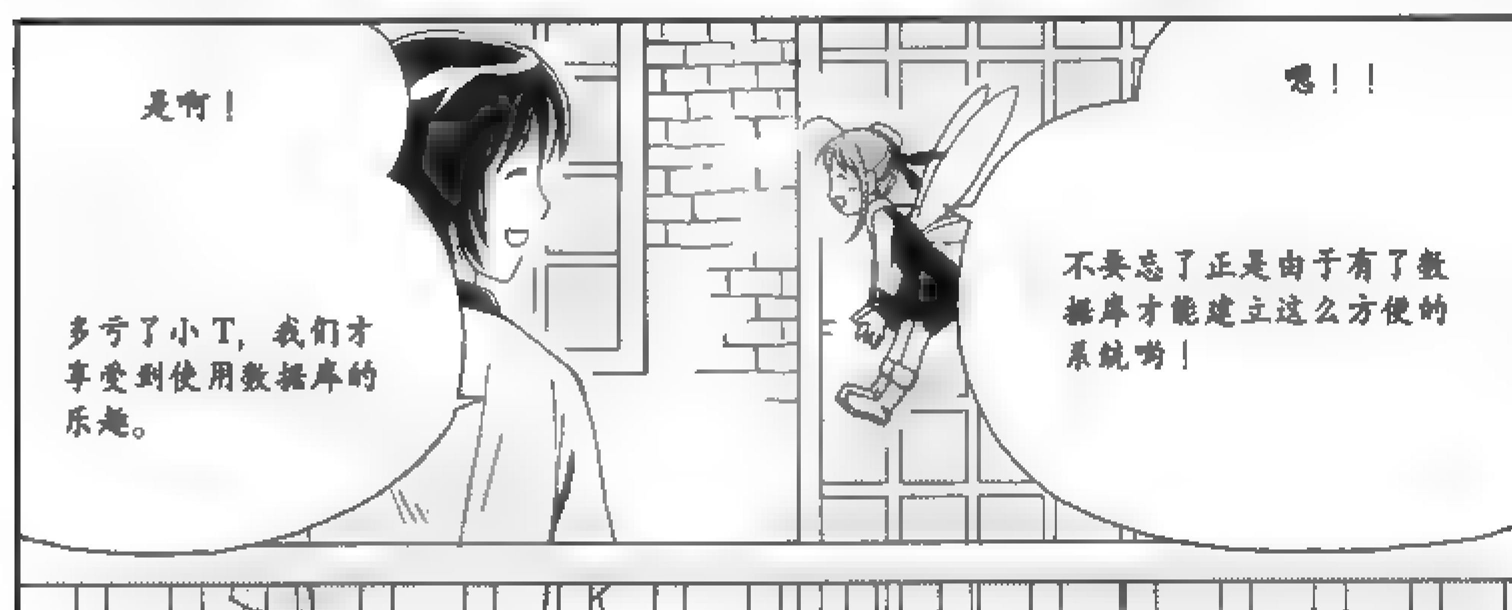
例如，
在数据更新的时候存储
程序就会自动启动，
这就叫做触发器（trigger）。
触发器……啊！
ぴょ


对了，
触发器就是扳机！
就像触发器（扳机）启动就会发射
子弹一样，数据一旦更新就会启动
存储程序。
西部牛仔

也就是说某人发出订单
更新了数据库时，
为什么我也变成
这副样子了？
能自动减少库存
进行配送处理的
话真是方便啊！

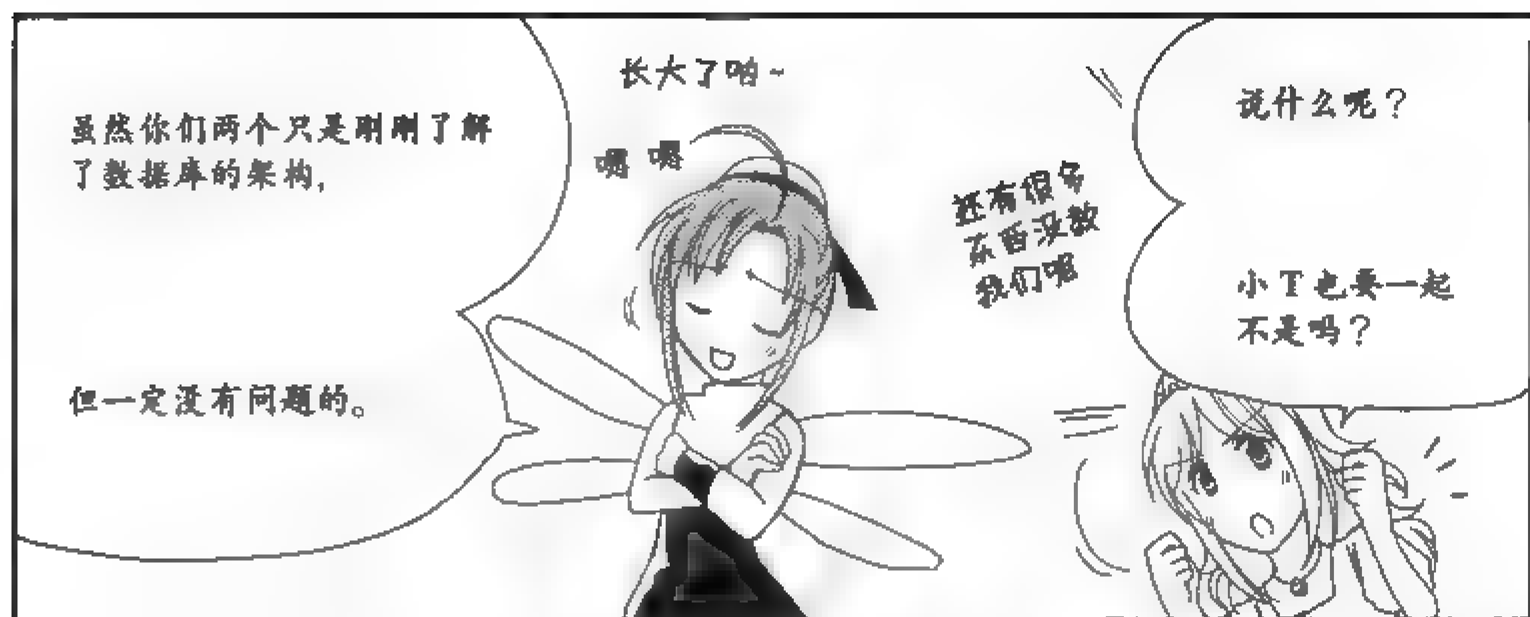
没想到买一本书还用
到了这么多功能呢。

嗯！
是啊，
真正在 Web 上买书的
时候，
好像很少能直接感觉
到数据库的存在。

如果我们没有一部分
一部分地好好学习的
话，还真是不知道呀！
是吧，凯恩！

是啊！
多亏了小 T，我们才
享受到使用数据库的
乐趣。
嗯！！
不要忘了正是由于有了数
据库才能建立这么方便的
系统哟！

好，编码王国要更好地
利用数据库，
成为一个人们生活便利
的国家。

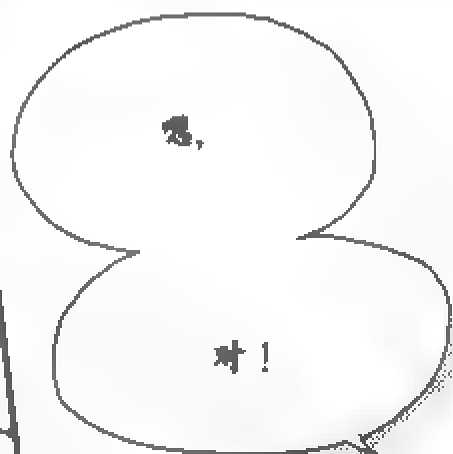
加油吧！！
嗯—!!


虽然你们两个只是刚刚了解了数据库的架构，
但一定没有问题的。
长大了呐~
嗯 嗯
还有很多东西没教我们呢
说什么呢？
小T也要一起不是吗？

很遗憾！
不能教你了。
我的职责就到这里了。

啊……
咔嗒
我得去找下一个人了。

因为有好多好多人想要学习数据库的知识呢。
那就是说要去下一个得到数据库这本书的人那里了？


嗯，
对！

因为我是……

数据库精灵！

びしっ

本来今天是要说再见的，

呵呵

结果又唠唠叨叨说了很多。

虽然相处时间很短，但却很开心！

小T……

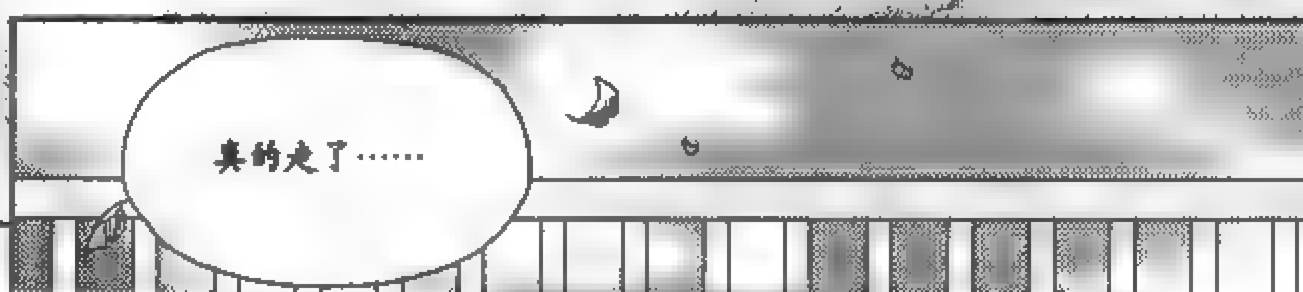
小T，等一下！！

我还没……

小T……

小T！！

谢谢！！

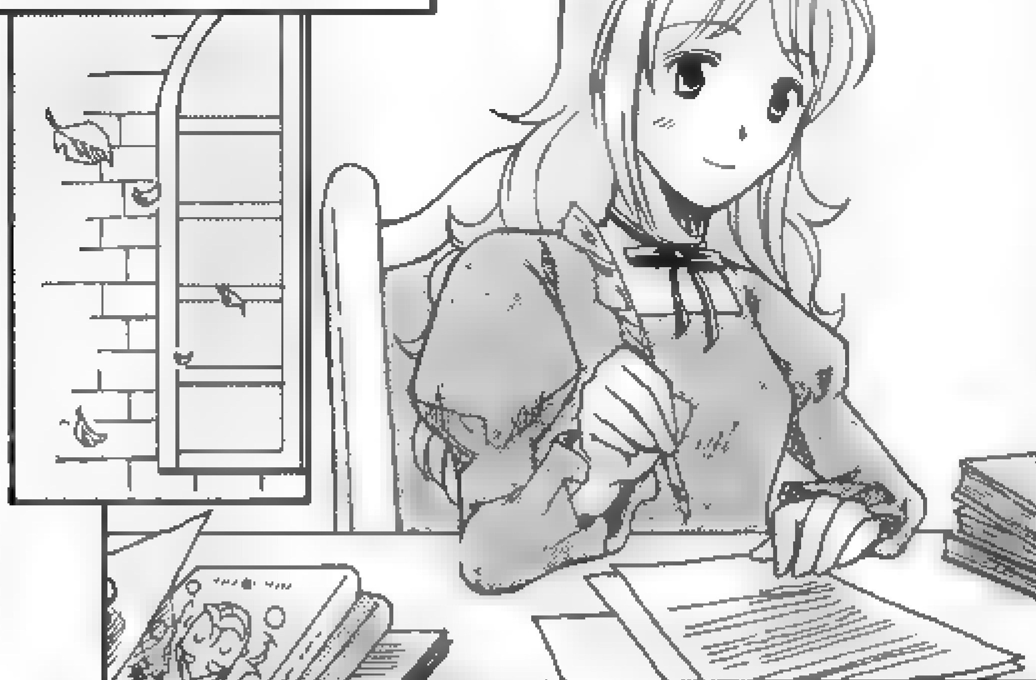


不要难过了！

小T教给了我们知识，

现在轮到我们好好使用了！

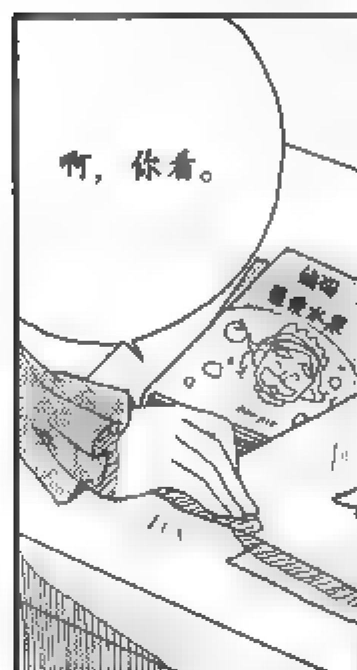
嗯！


后来——

还在写呀？
关于数据库的书？
呵呵

嗯！根据我的亲身体会，
想让大家更加容易理解！
看吗？
哇

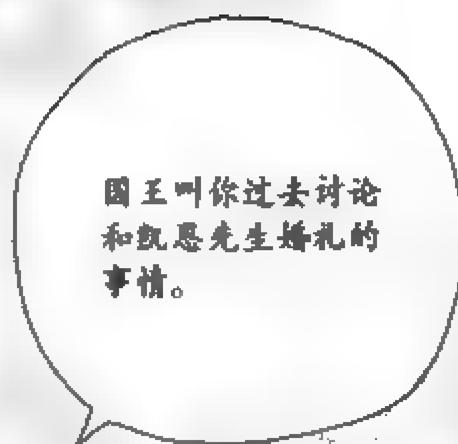
好可爱啊~
用图解的方式，来写真是个好主意，
凯恩先生的画真漂亮啊！


啊，你看。

这个是封面。

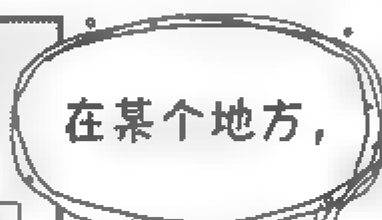
哇！

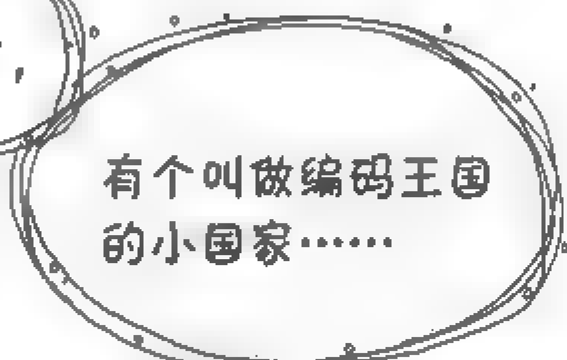
好漂亮！
真有意思~♡
对了……


国王叫你过去讨论和凯恩先生婚礼的事情。

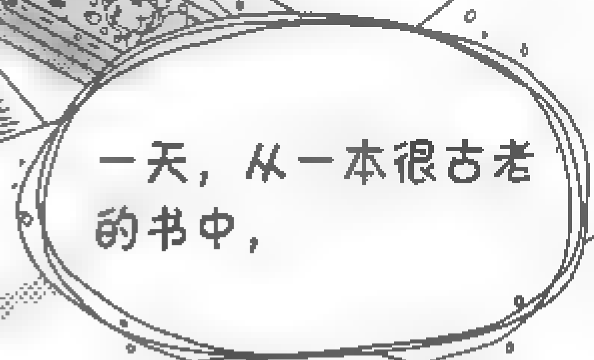
父亲？

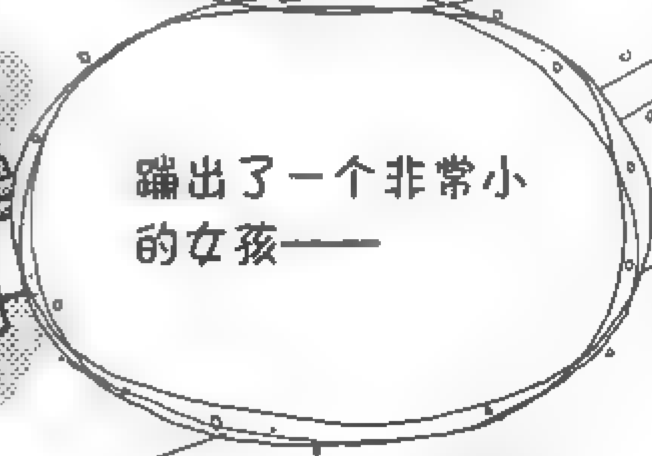
知道了，一起去吧！
好的
咔嚓


在某个地方，


有个叫做编码王国的小国家……

ぱたん..
编码王国的数据库
编码王国的数据库


一天，从一本很古老的书中，

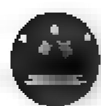

蹦出了一个非常小的女孩——

结局。

活跃的数据库

数据库的用途很广。例如，火车票预定系统和银行储蓄系统等都用到了数据库。在日常生活和公司经营中数据库已然成为不可或缺的工具。

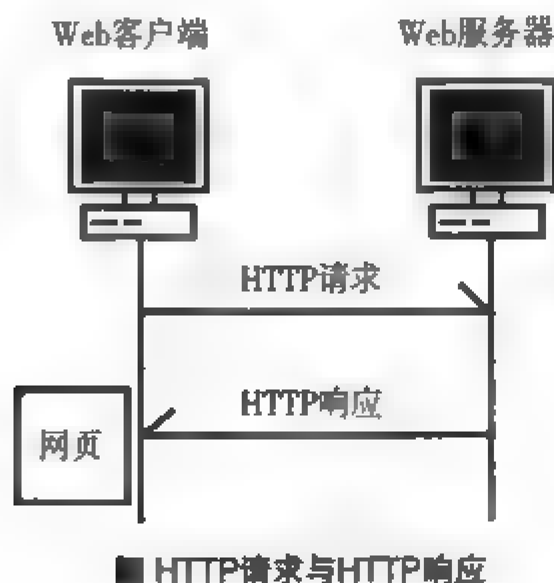
特别是现在，使用网络系统的数据库系统普及开来。小T也向露娜公主和凯恩介绍了使用网络的系统。



网络(Web)和数据库

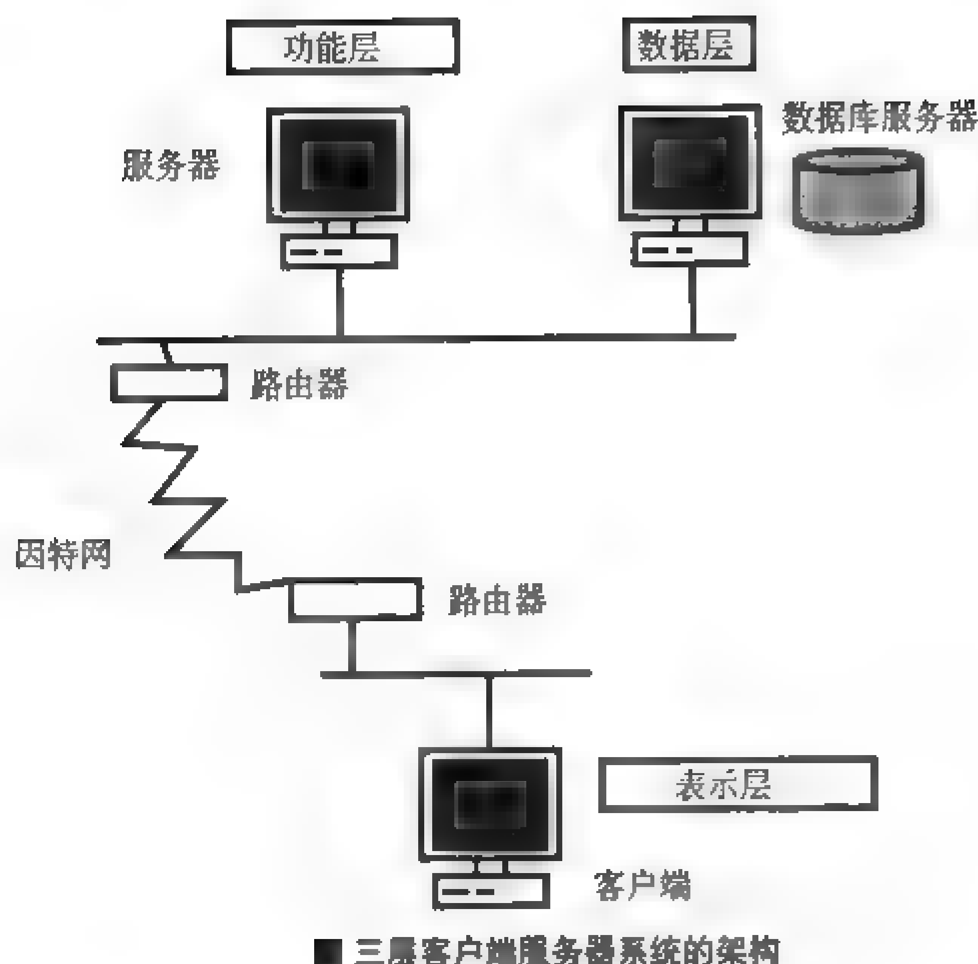
网络系统中使用通信协议 HTTP(HyperText Transfer Protocol)。Web 服务器上运行的服务器软件等待来自用户的请求。用户发出请求后，便会返回该请求所对应的网页 (HTTP response)。

网页由 HTML 格式的文本文件构成。网页中的图像等其他文件由 URL(Uniform Resource Locator) 指定嵌入。有嵌入文件时，访问指定的 URL，进行 HTTP 请求和响应处理。



■ HTTP请求与HTTP响应

使用数据库时，要向这样的系统中加入数据库服务器。目前，这个系统由三层构成。即所谓的三层客户端服务器系统。三层客户端服务器系统由表示层、功能层和数据层构成。

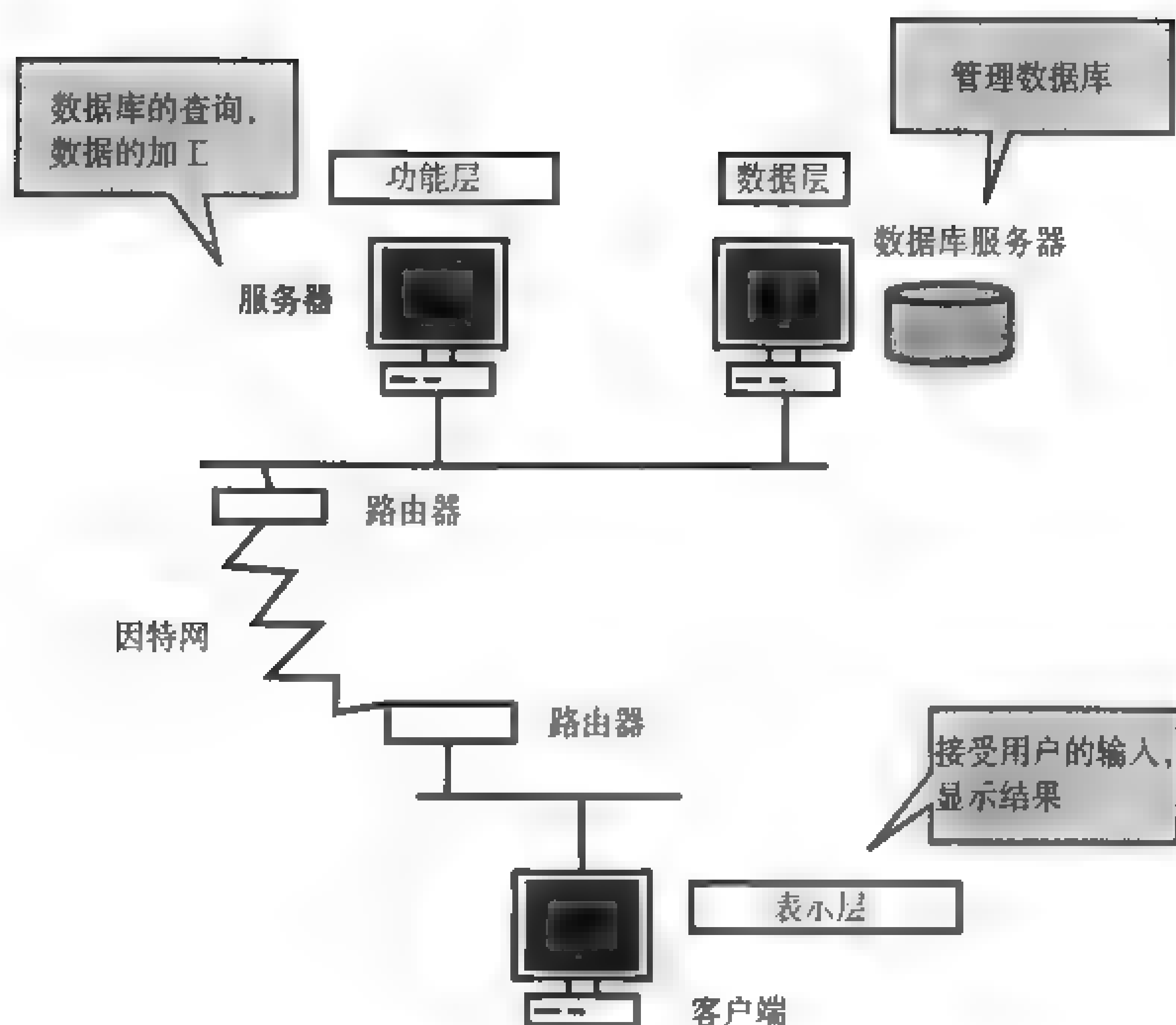


■ 三层客户端服务器系统的架构

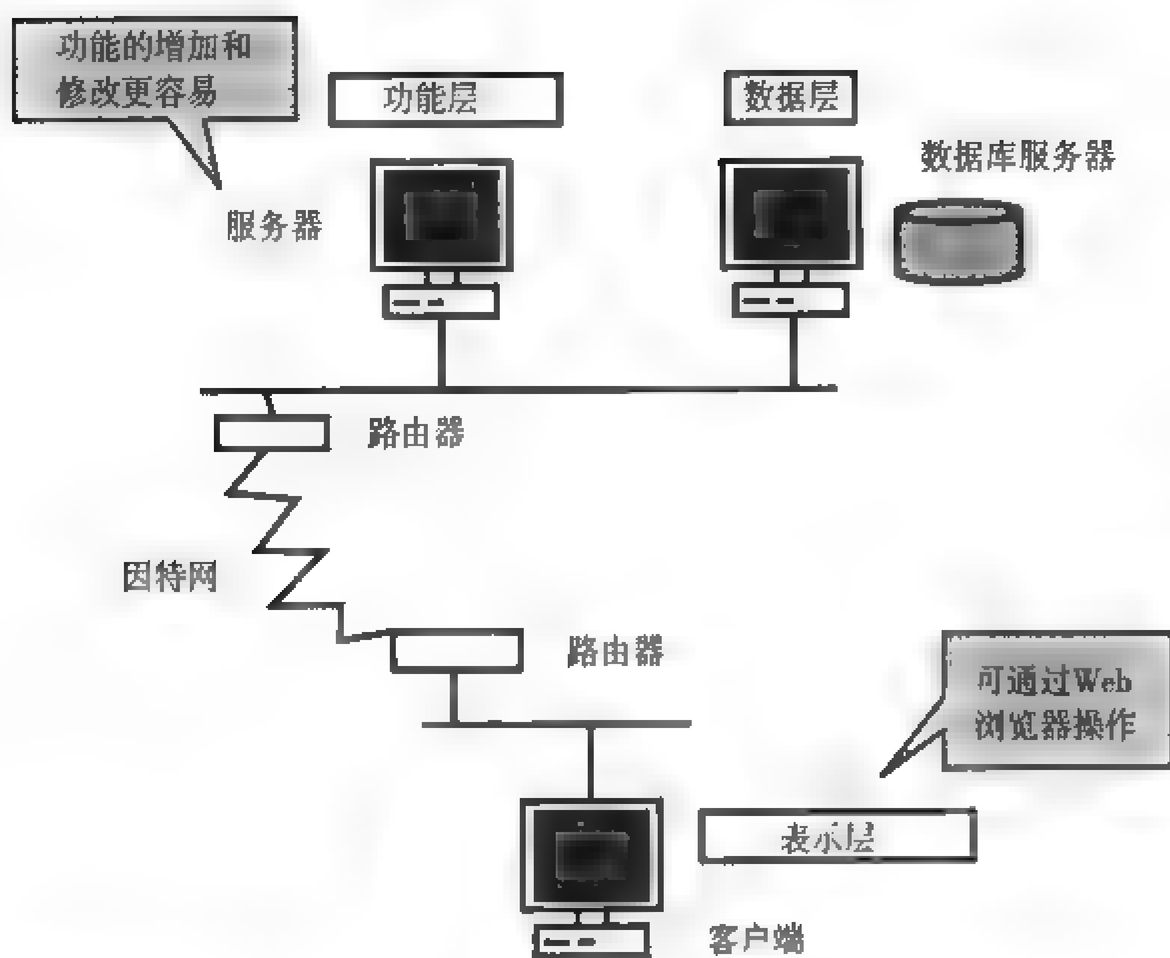
表示层接受用户的输入、获取对数据库的检索条件等。另外，表示层还会对数据库进行查询后获得的结果进行显示。通常由 Web 浏览器实现表示的功能。

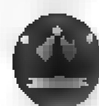
功能层进行数据加工。该层对 SQL 命令进行整合。这些处理由各种程序语言等记述。根据处理的内容和负荷，也可由应用服务器、Web 服务器等多个服务器分担处理。

数据层由数据库服务器处理。数据库服务器管理数据库。根据 SQL 等的查询要求，检索结果从数据库返回。



三层客户端·服务器型的结构能够构筑灵活简洁的系统。例如，想要在应用功能中进行很多添加、修改时，作为功能层由于将系统分割开来，使得开发变得更容易。另外，在表示层，由于使用了浏览器，用户不再需要重新制作、安装操作程序。



 使用存储程序

但是，在这样一个系统中，网络上流动数据的通信量还是个问题。数据库中由于有存储程序等功能，所以能够在数据库服务器中配置预先查询等程序。

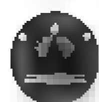
由于数据库服务器中配置了程序，因此没有必要频繁发送SQL查询命令，存储程序能够减少网络的负荷。另外，由于记述了定型的处理，应用的开发也变得更容易了。存储程序还有以下各种名称。

■ 存储程序的种类

存储过程	没有处理步骤返回值的程序
存储功能	返回从处理步骤来的返回值的程序
触发	在数据库中进行操作前后启动的程序

在三层客户端·服务器系统中，数据库运行的层叫什么?

在三层客户端·服务器系统中，接受用户的操作显示结果的层叫什么?



分布式数据库

在使用 Web 的系统中，由数据库服务器、Web 服务器、Web 浏览器等分担功能，分散处理系统。这种分布式的系统能够根据服务器的处理能力进行灵活的处理。

数据库本身也可能分散到别的环境。我们把这叫做分布式数据库 (distributed database)。分布式数据库是指由网络构成的、位置分散的数据库。

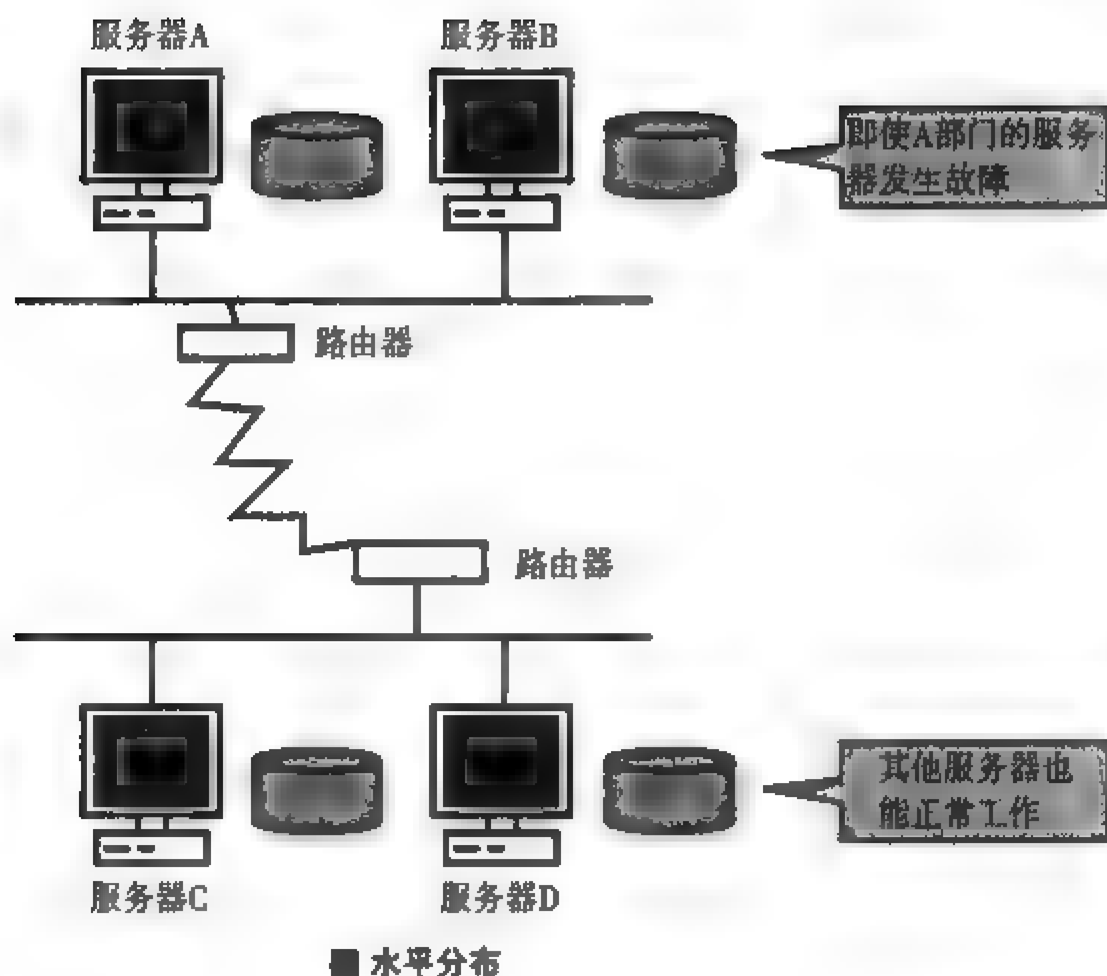
分布式数据库尽管被分散了，但是仍有必要把它当成一个数据库集中处理。用户可以不管数据的位置和移动使用分布式数据库这种特性，我们称之为透明度 (transparency)。

分布式数据库的构成方式分为水平分布和垂直分布两种。

水平分布

水平分布是指使用多个同级别的数据库服务器的方法。每个数据库服务器都能够使用别的数据库服务器的数据。相反，其他数据库服务器也可以使用本数据库的数据。这种形态多用于每个部门控制一个数据库扩大了的系统中。

水平分布式数据库中即便一个服务器发生故障，数据库仍能正常工作。对故障有很强的抵御功能。



■ 水平分布

垂直分布

垂直分布是数据库服务器具有不同功能的分布方式。由承担主要任务的主服务器和承担其他处理任务的服务器构成。各服务器可以使用主服务器的数据库，但是主服务器不使用其他服务器的数据。因此，在垂直分布式数据库中，主服务器易于管理的背后是主服务器集中承担通信任务。这种分布方式多用于数据库由整个组织的主服务器和各部门服务器构成的情况。

主服务器运行
主服务器
服务器A
路由器
路由器
服务器B
服务器C

■ 垂直分布

分配数据

分布式数据库的数据是分配给各服务器分别存储的。因此必须注意数据是如何被分配的。分配方法有以下几种。

水平分配方式

水平分配是将数据按照行的方向进行分割。分割后的行分别配置给各服务器。例如，相同类型的数据分地域管理时，就会采用这种方式。

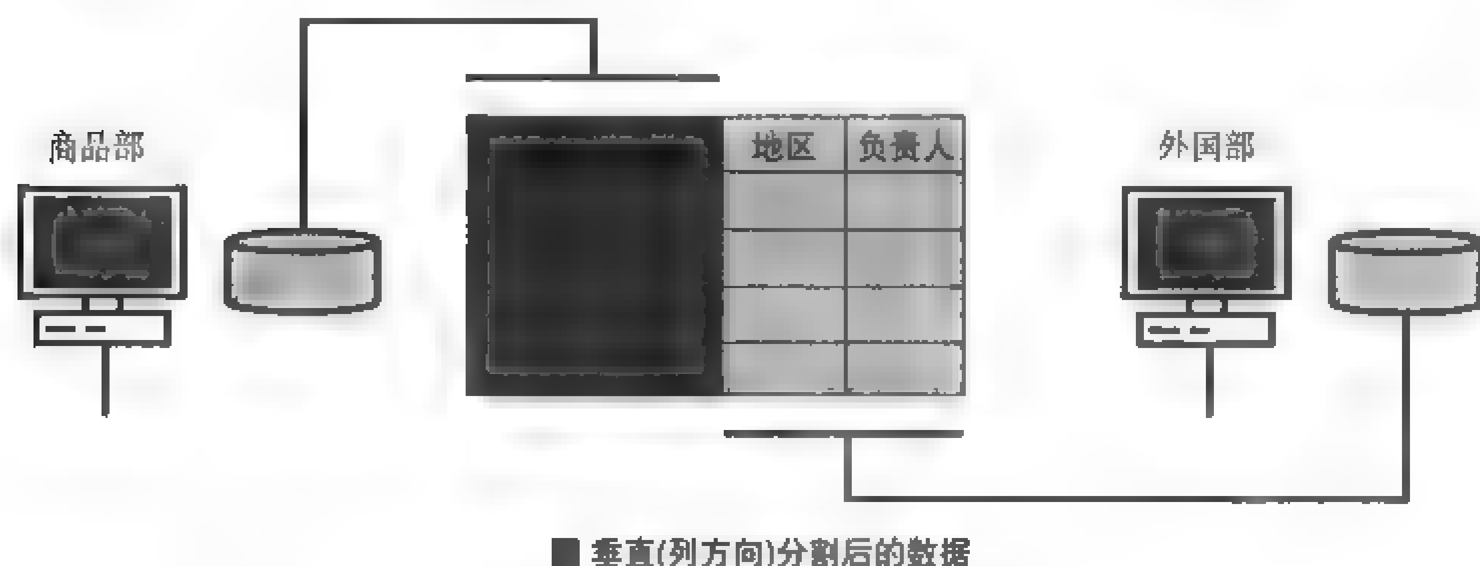
A地区营业部
B地区营业部

商品名称	单价	地区	负责人

■ 水平(行方向)分割后的数据

垂直分配方式

垂直分配是将数据按照列的方向进行分割。分割后的列分别配置给各个服务器。例如，在希望将商品部、出口部和外国部等独立部门的数据库按照分布式数据库进行管理时多采用这种方式。



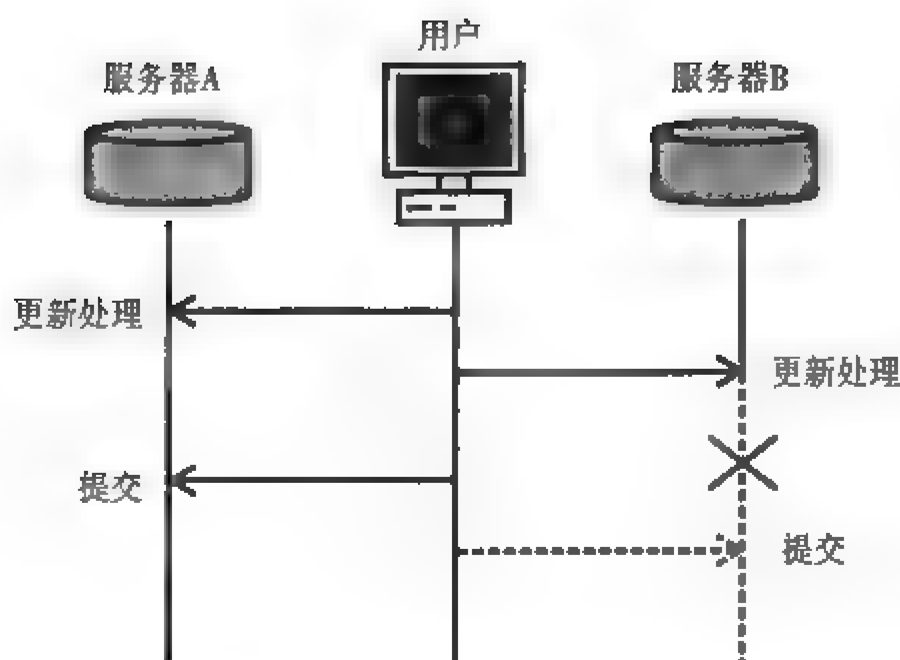
■ 垂直(列方向)分割后的数据

防止两阶段提交的矛盾

分布式数据库必须是可以让用户作为一整个数据库来处理的数据库。因为数据库分散在各个服务器，因此有很多需要注意的地方。

首先，数据被提交时不能与任何一个服务器的数据产生矛盾。

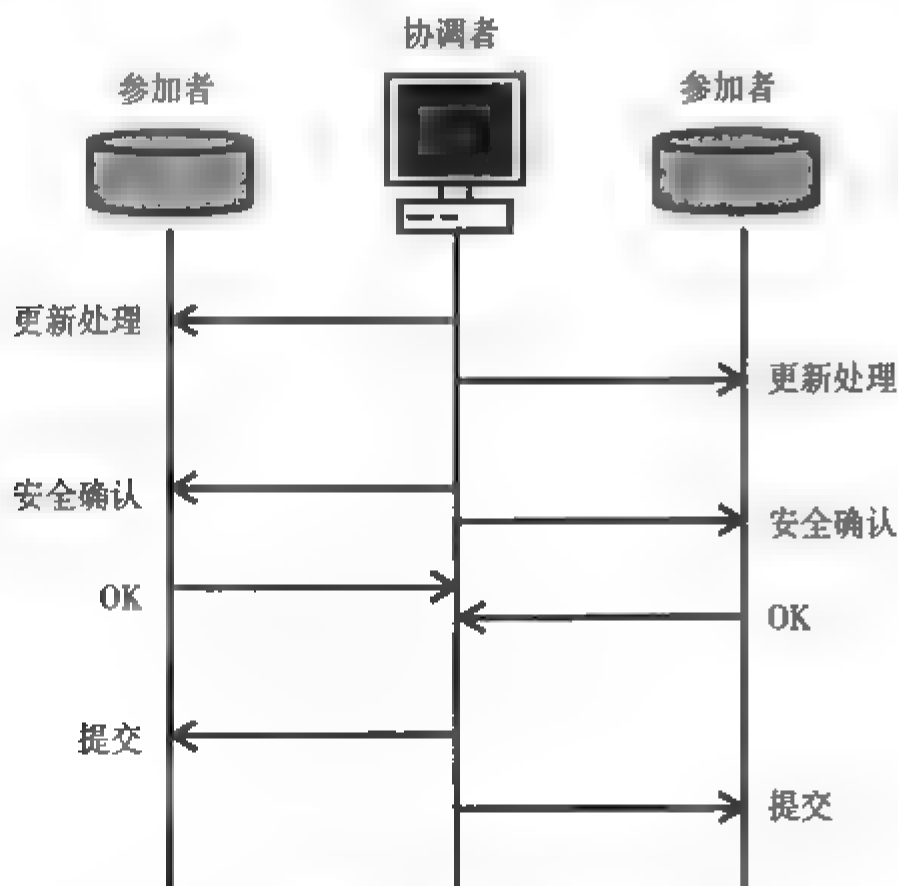
在分布式数据库中，按照下页图例所示的普通提交方式，有可能出现一个服务器数据更新，但另一个服务器的数据没有更新的情况。这样，就违反了事务的原子性，即一个事务必须由提交或回滚中的任意一个来结束。另外，作为整体的数据库发生矛盾，也违反了事务的一致性。



■ 一个服务器更新数据后产生矛盾

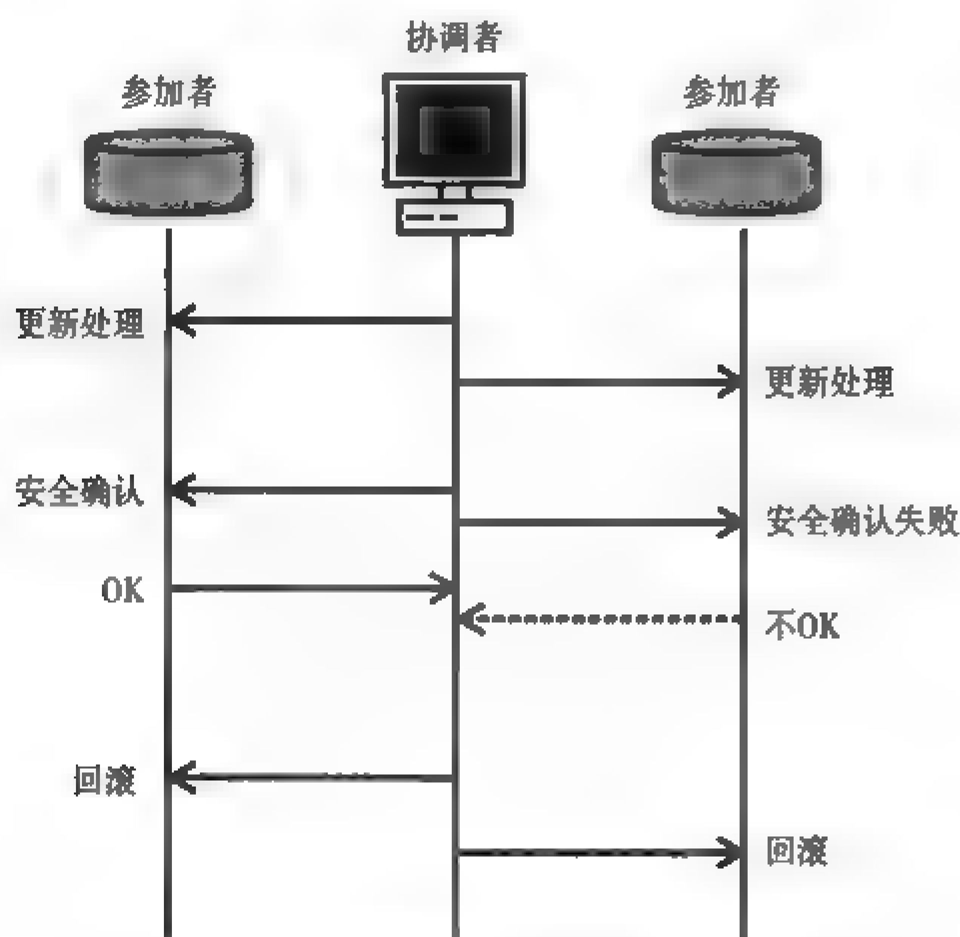
因此，分布式数据库采用名叫两阶段提交 (two-phase commit) 的方法。两阶段提交是一种提交程序由第一和第二两个阶段来构成的方法。

两阶段提交由协调者 (coordinator) 和参与者 (participant) 构成。在两阶段提交的第一阶段，协调者查询各参与者是否能够提交。参与者能够提交时，回复 OK。这个准备工作称为安全确认 (secure)。第二阶段，协调者发出提交指令，由各参加者进行提交。



■ 两阶段提交流程(提交完成的情形)

在两阶段提交过程中，第一阶段一个参与者安全确认失败，将会对所有参与者发出回滚的指令。因此，这就形成了确保各服务器的数据库不产生矛盾的架构。



■ 两阶段提交流程(安全确认失败，回滚的情形)

在两阶段提交中，协调者在第一阶段发出什么指令？

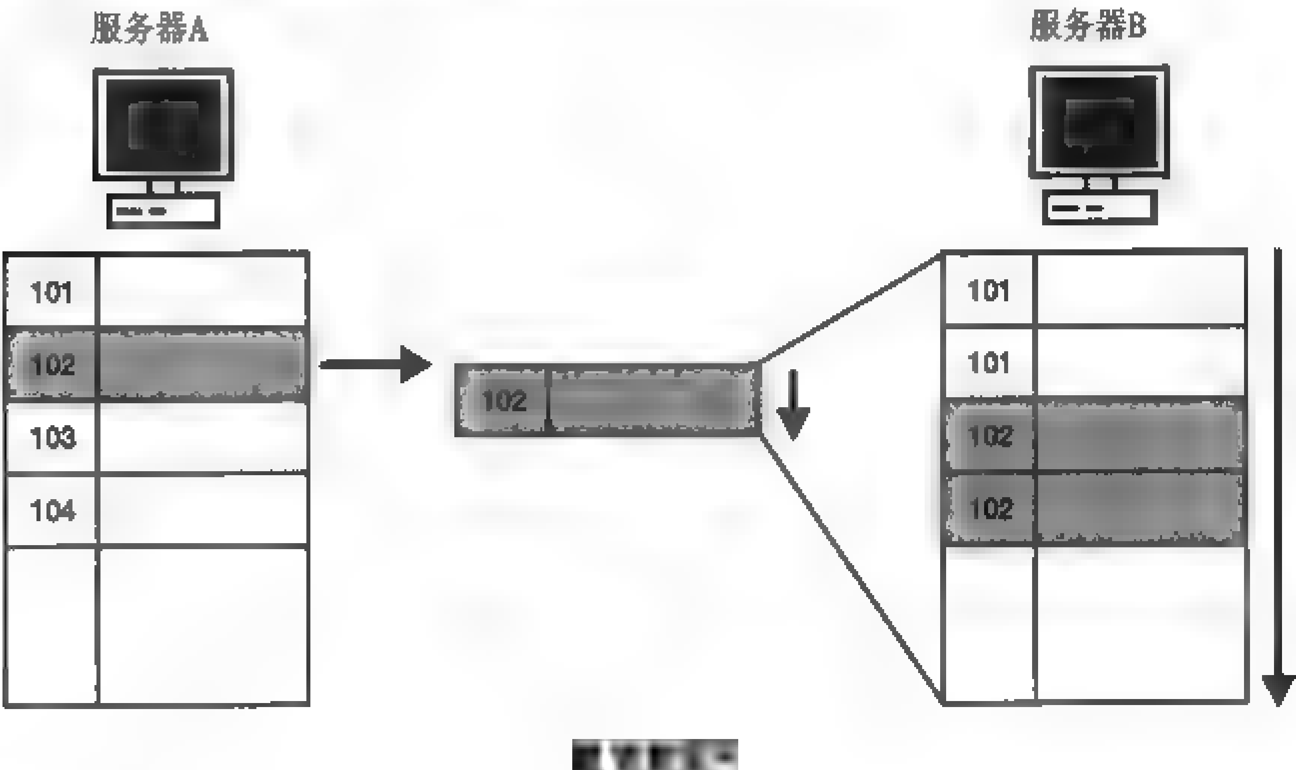
在两阶段提交中，协调者在第二阶段发出什么指令？

分布式数据库中表格的连接

分布式数据库增加了网络间的通信量，从而增大了网络的负荷。特别是在服务器之间连接表格的情况下，必须注意数据的通信量。分布式数据库连接表格的方法有以下几种：

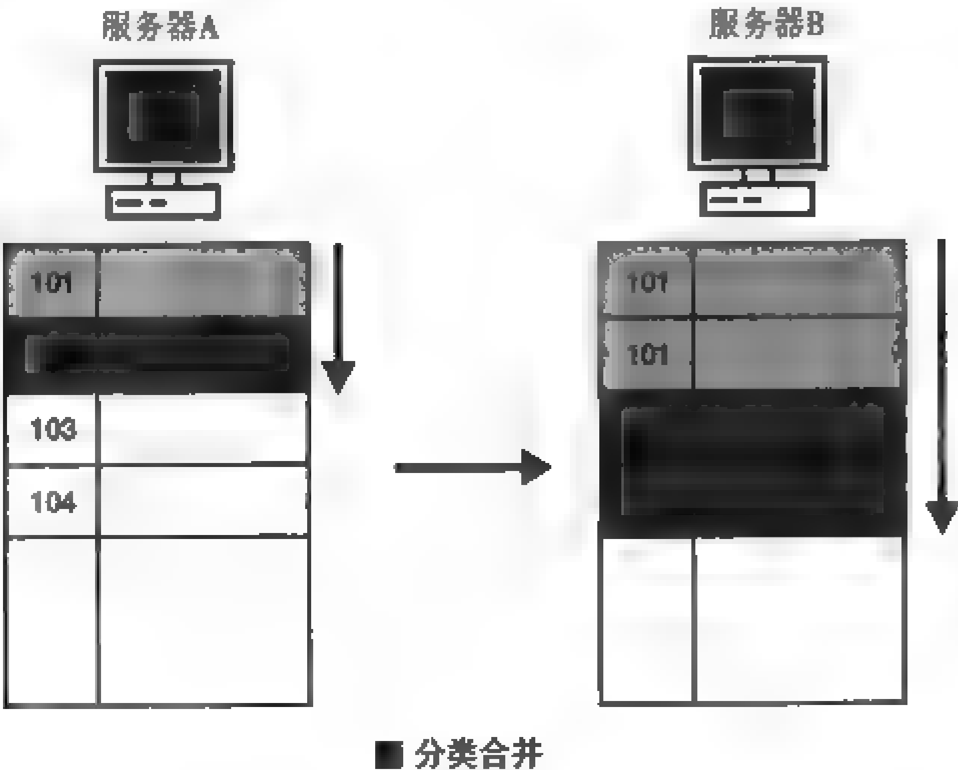
嵌套循环 (nested loop)

将服务器 A 中表格的某一行发送至服务器 B，与服务器 B 中表格的每一行进行比较之后连接。将服务器 A 中表格的每一行都按照此程序反复操作。



分类合并 (sort merge)

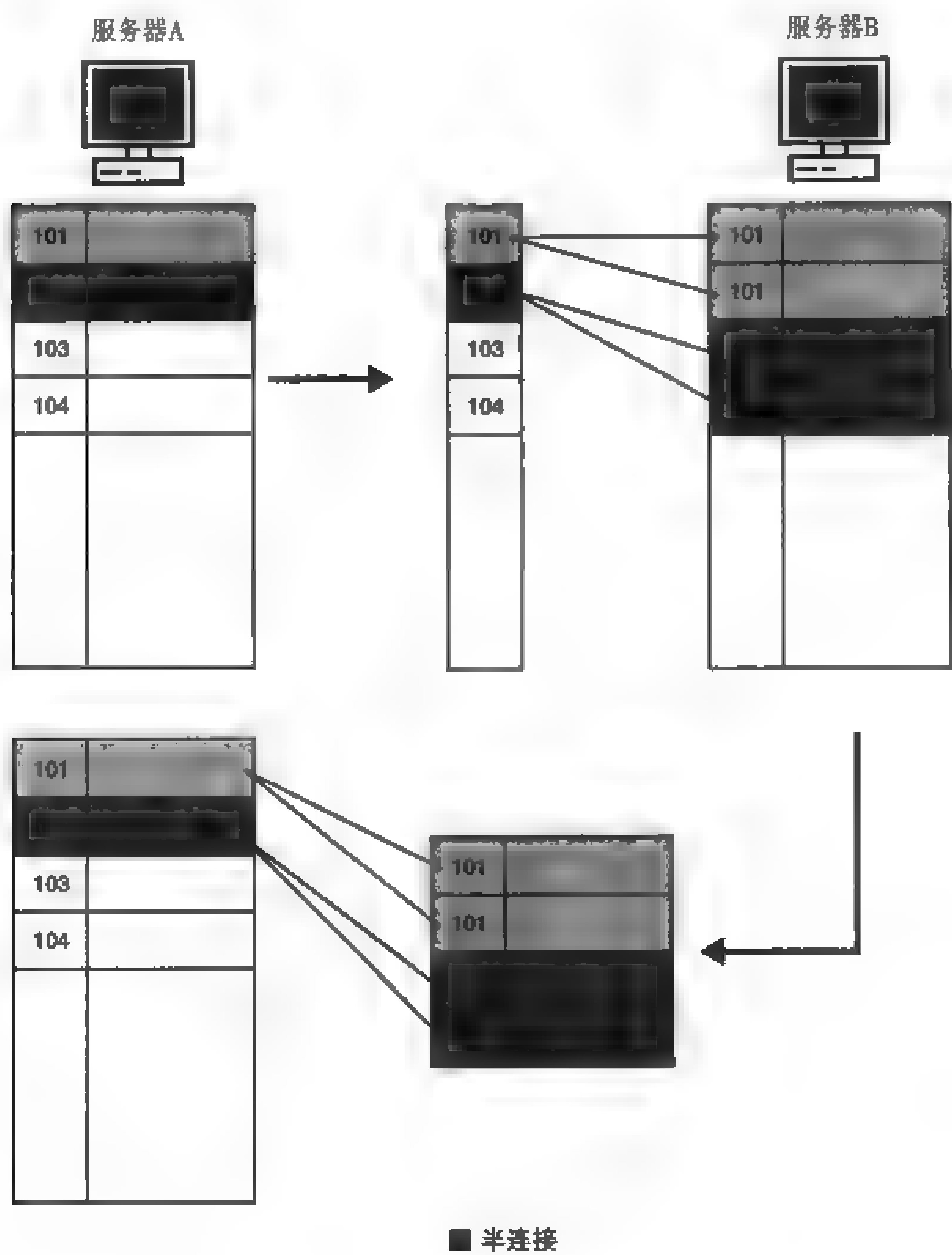
分类合并是一种将各服务器表格预先分类的方法。将服务器 A 的表格和服务器 B 的表格各自分类。之后将服务器 A 的表格发送至服务器 B。因为表格事前已经分类，因此能够通过一个方向的读取进行连接。



■ 分类合并

半连接 (semi join)

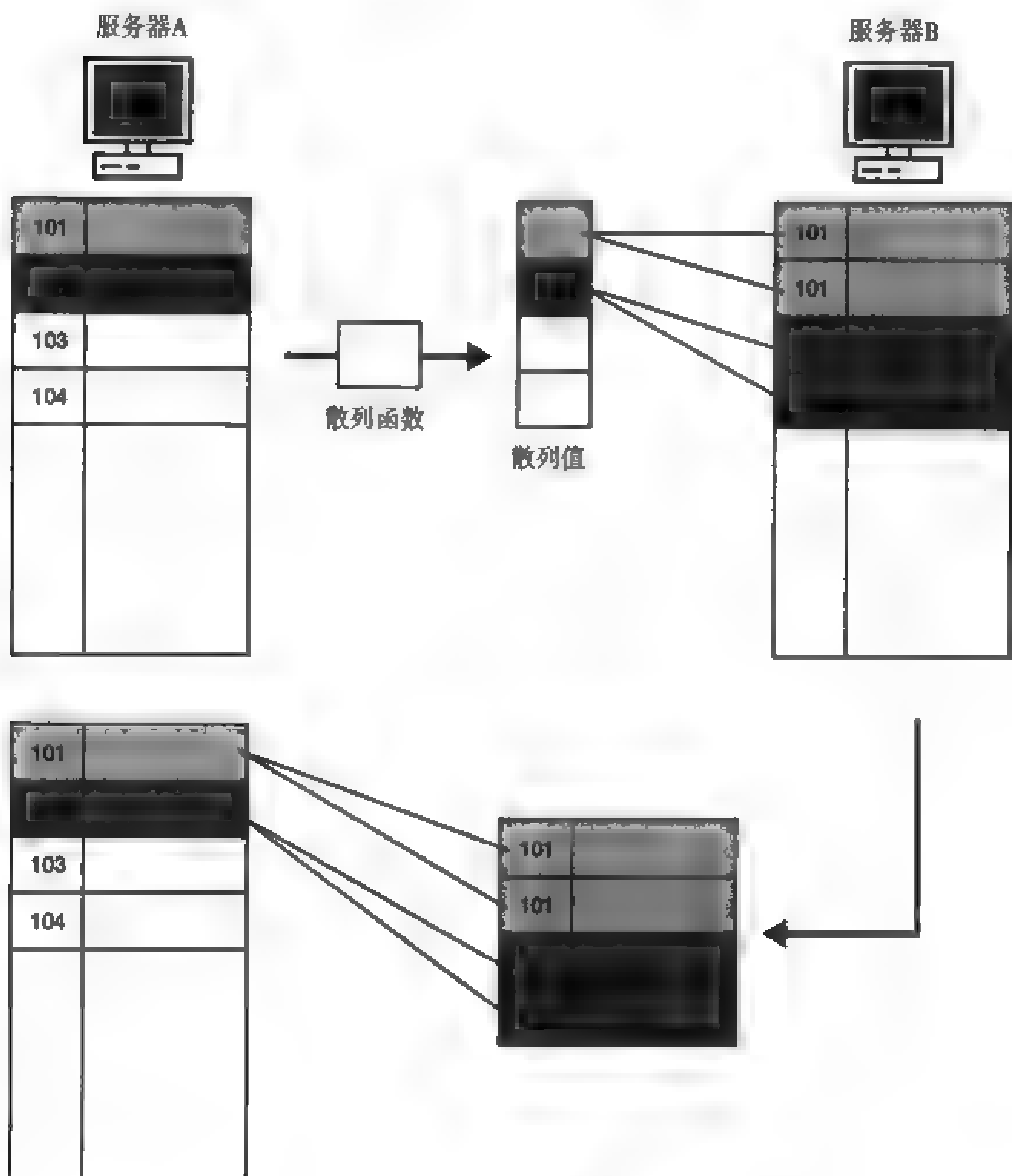
半连接是一种仅将与连接相关的列发送至与之连接的服务器，在减少行之后进行连接的方法。例如，首先将服务器 A 的商品编码列发送至服务器 B。之后抽取服务器 B 中相应的商品编码。再将抽取的行发送回服务器 A，根据这些行进行连接处理。因为减少了行，所以可能减少了网络的通信量。

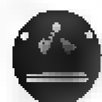


■ 半连接

散列半连接 (hash semi join)

在散列半连接过程中，先求取服务器 A 中列的散列值，并发送至服务器 B。在服务器 B 中也求取散列值。通过散列值之间的检查进行连接。



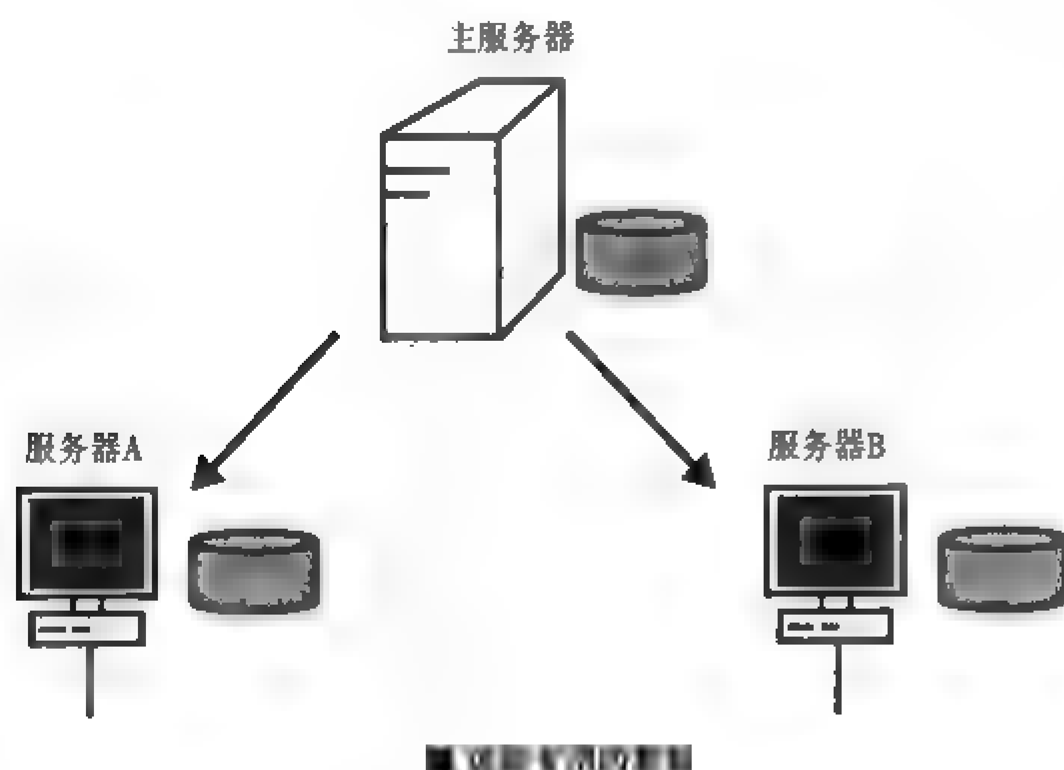


复制的配置

在分布式数据库中，为了减少网络的负荷，数据库设置了复制的功能，我们称之为复制。通过重复使用数据的复制，减少了网络中传送的数据。我们将主要的数据库称为主数据库。复制称为 replica，复制的方式有以下几种。

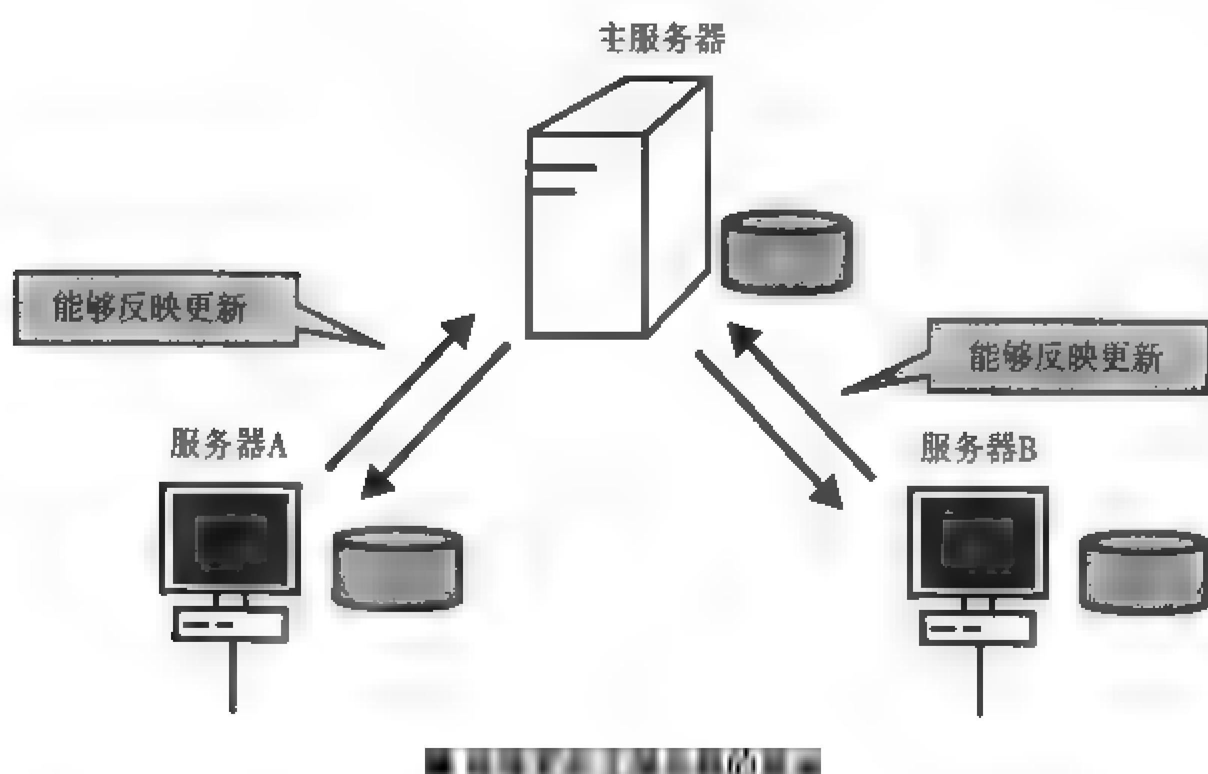
读取专用

是一种生成并下载从主服务器的主数据库中读取专用复制的方法。复制是在主服务器连接时生成的。复制仅可读取。



可更新主服务器

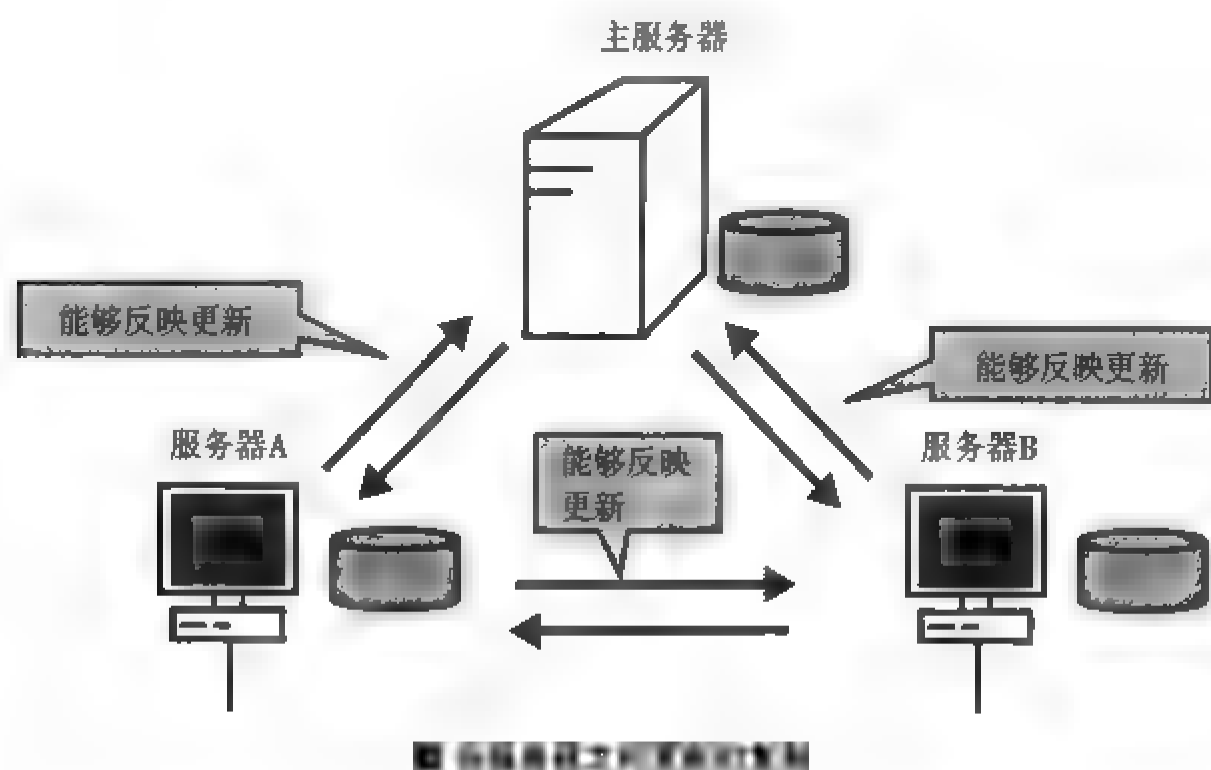
从主服务器生成复制。该复制能够更新。更新复制时被反映在主服务器的主数据库中。



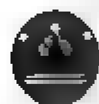
[illegible]

可更新各个服务器

各服务器之间拥有相同主服务器的方法。在各个服务器中更新后，可反映在其他服务器的数据库中。



[illegible]



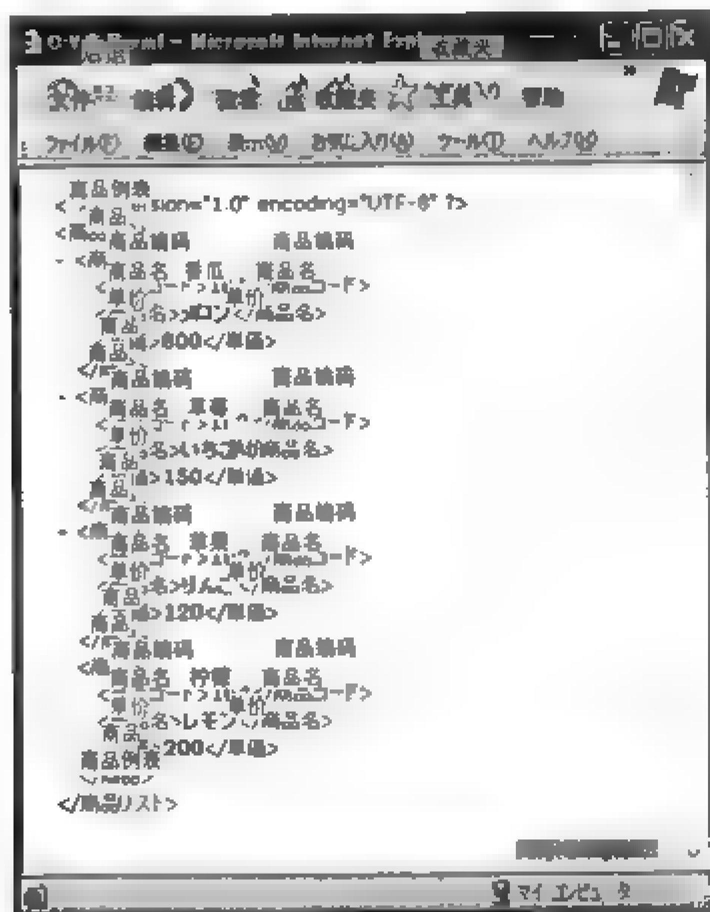
数据库的深层次应用

最后我们来介绍一下数据库相关应用技术。

XML

作为数据的存储方式，XML(Extensible Markup Language) 倍受瞩目。XML 是一种用标签区分数据的可扩展标记语言。由于能够用标签赋予数据独特的含义，因此在检索方面功能非常强大。

XML 拥有非常严密的语法，具有非常易用于程序处理的特征。并且，由于 XML 是文本文件，在进行编辑以及在不同系统间进行通信也就非常容易。因此经常在数据库中作为替代性的数据表现手段。



■ XML文档示例

面向对象数据库 (OODB)

关系数据库会将不同字符的数据以表格的形式进行存储。但是，在处理图像等数据时，关系数据库就比较缺乏灵活性了。因此，就要使用引入面向对象概念的数据库。

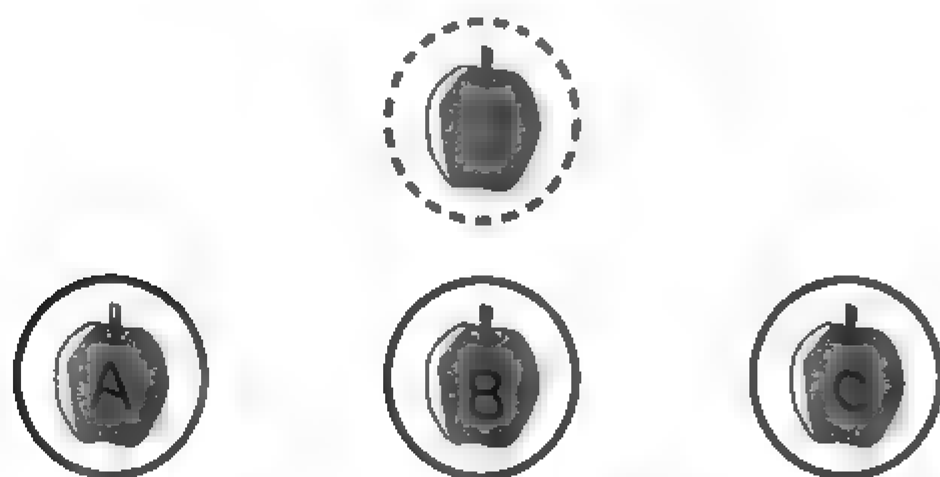
面向对象 (object oriented) 是建立在“对象”概念上的方法学。对象是指数据及其操作方法的一个综合体。它是通过隐藏数据仅公布操作方法，将对象作为一个牢固的部件进行处理的方法。这种功能叫做封装 (encapsulation)。

在面向对象数据库中对每个对象都添加标识符进行显示。对象称作事例 (Instance)。

在面向对象数据库中，能够在对象中嵌入对象，进行复合对象管理。例如，可以将一个集图像和文本作为一个信息对象，原封不动地存进数据库。通过面向对象数据库，我们可以对比字符复杂的数据进行灵活的管理。

■ [illegible]

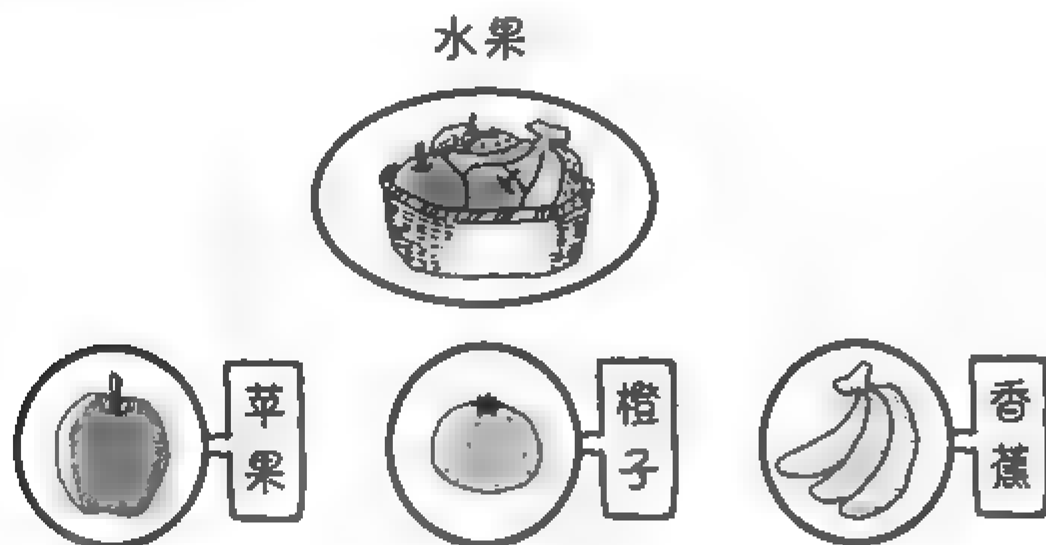
面向对象中也准备了实现高效开发的概念。首先，对象类型的概念叫做类 (class)。例如设置了“苹果”类这一概念后，苹果 A、苹果 B……都可以作为对象 (事例) 来对待。基于苹果类，也就能够生产成苹果 A、苹果 B。



■ 从类生成对象

另外在面向对象中，类也可以拥有阶层关系。能够继承上一层类的数据和功能，生成具有独特功能的下一层类。这种关系我们称之为继承(inheritance)。

例如，继承“水果”类概念可以生成“苹果”类、“橙子”类等。在面向对象中使用这一功能，可以进行高效的开发。



■ 能够设计继承功能的类

小结

- 构成 Web 系统的方法——三层客户端・服务器系统。
- [illegible]
- 分布式数据库处理分散了的数据库。
- 两阶段提交用于分布式数据库。

答案

A1 数据层

A2 表示层

A3 安全确认

A4 提交或回滚

结　语

数据库知识学习得怎么样？在实际设计和使用数据库的过程中，我们还需要更多的知识。

但是，数据库的基本概念都是一样的。目前我们所学到的知识中的形式、名称也有可能发生变化。按照基本架构能够从现实世界中发掘数据，设计并使用数据库。基于基础知识，希望你能掌握更深层次的数据库技术。

附录　常用SQL命令

●基本检索

```
SELECT 列名，…
FROM 表格名称
WHERE 条件；
```

●模糊检索

```
SELECT 列名，…
FROM 表格名称
WHERE 列名 LIKE “样式”；
```

●排列

```
SELECT 列名，…
FROM 表格名称
WHERE 条件
ORDER BY 列名；
```

●合计 分组化

```
SELECT 列名，…
FROM 表格名称
WHERE 条件
GROUP BY 分组列名
HAVING 分组行的条件；
```

●表格的连接

```
SELECT 列名，…
FROM 表格名称1，表格名称2，…
WHERE 表格名称1.列名＝表格名称2.列名；
```

●生成基本表

```
CREATE TABLE 表格名称(
列的定义
……
)；
```

●生成可视表

```
CREATE VIEW 可视表名称
AS SELECT命令；
```

●删除基本表

```
DROP TABLE 表格名称；
```

●删除可视表

```
DROP VIEW 可视表名称；
```

●插入行

```
INSERT INTO 表格名称(列名，……)
VALUES(值，……)；
```

●更新行

```
UPDATE 表格名称
SET 列名＝值，……
WHERE 条件；
```

●删除行

```
DELETE FROM 表格名称
WHERE 条件；
```

参考文献

- 石畑清、アルゴリズムとデータ構造、岩波書店、1989
- 増永良文、リレーショナルデータベースの基礎、オーム社、1990
- 飯沢篤志他、データベースおもしろ講座、共立出版、1993
- 北川博之、データベースシステム、昭晃堂、1996
- C.J. Date他、標準SQLガイド改訂第4版、アスキー、1999
- 情報処理技術者試験センター、情報処理技術者スキル標準テクニカルエンジニア（データベース）、2000
- E.F. Codd, A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, Vol.13, No.6, pp. 377-387, 1970
- P.P. Chen, The Entity-Relationship Model: Toward a Unified View of Data, ACM Transactions on Database systems,Vol.1, No.1, pp.9-36, 1976
- ISO/IEC 9075, Information Technology－Database Languages－ SQL, 1992
- ISO/IEC 9075, Information Technology－Database Languages－ SQL, 1995
- ISO/IEC 9075－1, 2, 3, 4, Information Technology－ Database Languages－ SQL, 1999
- データベース言語SQL, JIS X3005-1～4, 2002

？
B
O
嘣
请多关照！
じゃ～ん♪
什，什么东西！？
咦？好像在
哪里见过？
妖怪？！

（TP-4658.0102）
责任编辑：唐　璐　赵丽艳
责任制作：董立颖　魏　谨
封面制作：许思麒

用漫画这种形式讲数学、物理和统计学，十分有利于在广大青少年中普及科学知识。

周恩来、邓颖超秘书，周恩来邓颖超纪念馆顾问
中日友好协会理事，《数理天地》顾问，全国政协原副秘书长

用漫画和说故事的形式讲数学，使面貌冷峻的数学变得亲切、生动、有趣，使学习数学变得容易，这对于提高全民的数学水平无疑是功德无量的事。

《数理天地》杂志社　社长 总编
“希望杯”全国数学邀请赛组委会 命题委员会主任

用漫画的形式，讲解日常生活中的数学、物理知识，更能让大家感受到数学殿堂的奥妙与乐趣。

《光明日报》 原副总编辑
中华炎黄文化研究会 常务副会长

科学漫画是帮助学习文科的人们用形象思维的方式掌握自然科学的金钥匙。

中国人民大学外语学院日语专业 主任
大学日语教学研究会 会长

在日本留学的时候，我在电车上几乎每次都能看到很多年轻的白领看这套图书，经济实惠、图文并茂、浅显易懂，相信这套图书的中文版也一定会成为白领们的手中爱物。

大连理工大学 能源与动力学院 博士 副教授

我非常希望能够在书店里看到这样的书：有人物形象、有卡通图、有故事情节，当然最重要的还有深厚的理工科底蕴。我想这样的书一定可以大大提升孩子们的学习兴趣，降低他们对于高深的理工科知识的恐惧感。

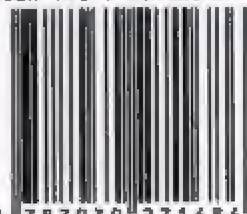
北京启明星培训学校 校长

书中的数学知识浅显实用，漫画故事的形式使知识贴近生活，概念更容易理解。

北京大学 数学科学学院 博士

上架建议：科普/漫画

ISBN 978-7-03-027168-6



科学出版社 东方科龙
http://www.okbook.com.cn
zhaoliyan@mail.sciencep.com

定价：32.00元